

L4Re – L4 Runtime Environment

Generated by Doxygen 1.8.13

Contents

1	Overview	1
2	Introduction	3
2.1	Fiasco.OC	3
2.1.1	Communication	3
2.1.2	Kernel Objects	4
2.2	L4Re System Structure	4
2.3	L4 Runtime Environment (L4Re)	6
3	Tutorial	7
3.1	Customizations	8
4	Programming for L4Re	9
4.1	Capabilities and Naming	9
4.2	Initial Environment and Application Bootstrapping	10
4.2.1	Configuring an application before startup	11
4.2.2	Connecting clients and servers	11
4.3	Memory management - Data Spaces and the Region Map	12
4.3.1	User-level paging	12
4.3.1.1	Data spaces	12
4.3.1.2	Virtual Memory Handling	12
4.3.1.3	Memory Allocation	13
4.4	Program Input and Output	13
4.5	Initial Memory Allocator and Factory	13
4.6	Application and Server Building Blocks	13

4.6.1	Creating Additional Application Threads	14
4.6.2	Providing a Service	14
4.7	Pthread Support	14
4.8	Interface Definition Language	15
4.8.1	Parameter types for RPC	15
4.8.2	RPC Return Types	17
4.8.3	RPC Method Declaration	18
4.9	L4Re Build System	18
4.9.1	Building L4Re	19
4.9.2	Writing BID Make Files	20
4.9.3	prog.mk - Application Role	20
4.9.4	include.mk - Header File Role	23
4.9.5	test.mk - Test Application Role	24
5	L4Re Servers	29
5.1	Moe, the Root-Task	31
5.1.1	Memory Allocator, Generic Factory	31
5.1.2	Name-Space Provider	32
5.1.2.1	Boot FS	32
5.1.3	Log Subsystem	32
5.1.4	Scheduler Subsystem	32
5.1.5	Command-Line Options	32
5.2	Ned, the Init Process	33
5.2.1	Lua Bindings for L4Re	33
5.2.1.1	Capabilities in Lua	33
5.2.1.2	Access to L4Re::Env Capabilities	33
5.2.1.3	Constants	34
5.2.1.4	Application Startup Details	35
5.2.1.5	Access to the kernel debugger	35
5.2.2	Command Line Options	36
5.3	Io, the Io Server	36
5.4	Uvmm, the virtual machine monitor	41
5.5	Mag, the GUI Multiplexer	44
5.6	Cons, the Console Multiplexer	46

6	Deprecated List	47
7	Module Index	49
7.1	Modules	49
8	Namespace Index	53
8.1	Namespace List	53
9	Hierarchical Index	55
9.1	Class Hierarchy	55
10	Data Structure Index	65
10.1	Data Structures	65
11	File Index	79
11.1	File List	79
12	Module Documentation	91
12.1	ARM Virtual Registers (UTCB)	91
12.1.1	Detailed Description	91
12.2	Atomic Instructions	92
12.2.1	Detailed Description	94
12.2.2	Function Documentation	94
12.2.2.1	l4util_add8()	94
12.2.2.2	l4util_add8_res()	94
12.2.2.3	l4util_atomic_add()	94
12.2.2.4	l4util_atomic_inc()	95
12.2.2.5	l4util_cmpxchg()	95
12.2.2.6	l4util_cmpxchg16()	96
12.2.2.7	l4util_cmpxchg32()	97
12.2.2.8	l4util_cmpxchg64()	98
12.2.2.9	l4util_cmpxchg8()	98
12.2.2.10	l4util_inc8()	99
12.2.2.11	l4util_inc8_res()	99

12.2.2.12	<code>l4util_xchg()</code>	100
12.2.2.13	<code>l4util_xchg16()</code>	100
12.2.2.14	<code>l4util_xchg32()</code>	100
12.2.2.15	<code>l4util_xchg8()</code>	101
12.3	Auxiliary data	102
12.3.1	Detailed Description	102
12.4	Base API	103
12.4.1	Detailed Description	105
12.5	Basic Macros	106
12.5.1	Detailed Description	107
12.5.2	Macro Definition Documentation	107
12.5.2.1	<code>L4_ALWAYS_INLINE</code>	107
12.5.2.2	<code>L4_HIDDEN</code>	107
12.5.2.3	<code>L4_NOTHROW</code>	108
12.6	Bit Manipulation	109
12.6.1	Detailed Description	109
12.6.2	Function Documentation	109
12.6.2.1	<code>l4util_bsf()</code>	110
12.6.2.2	<code>l4util_bsr()</code>	110
12.6.2.3	<code>l4util_btc()</code>	111
12.6.2.4	<code>l4util_btr()</code>	111
12.6.2.5	<code>l4util_bts()</code>	112
12.6.2.6	<code>l4util_clear_bit()</code>	113
12.6.2.7	<code>l4util_complement_bit()</code>	113
12.6.2.8	<code>l4util_find_first_set_bit()</code>	114
12.6.2.9	<code>l4util_find_first_zero_bit()</code>	114
12.6.2.10	<code>l4util_next_power2()</code>	115
12.6.2.11	<code>l4util_set_bit()</code>	115
12.6.2.12	<code>l4util_test_bit()</code>	116
12.7	Buffer Registers (BRs)	117

12.7.1 Detailed Description	117
12.7.2 Enumeration Type Documentation	117
12.7.2.1 l4_buffer_desc_consts_t	117
12.8 C++ Exceptions	119
12.8.1 Detailed Description	119
12.9 C++ IPC Interface Definition.	120
12.9.1 Detailed Description	120
12.10 CPU related functions	121
12.10.1 Detailed Description	121
12.10.2 Function Documentation	121
12.10.2.1 l4util_cpu_capabilities()	121
12.10.2.2 l4util_cpu_capabilities_nocheck()	122
12.10.2.3 l4util_cpu_has_cpuid()	123
12.11 Cache Consistency	124
12.11.1 Detailed Description	124
12.11.2 Function Documentation	124
12.11.2.1 l4_cache_clean_data()	124
12.11.2.2 l4_cache_coherent()	125
12.11.2.3 l4_cache_dma_coherent()	125
12.11.2.4 l4_cache_flush_data()	126
12.11.2.5 l4_cache_inv_data()	126
12.12 Capabilities	128
12.12.1 Detailed Description	128
12.12.2 Enumeration Type Documentation	129
12.12.2.1 l4_cap_consts_t	129
12.12.2.2 l4_default_caps_t	129
12.12.3 Function Documentation	130
12.12.3.1 l4_capability_equal()	130
12.12.3.2 l4_is_invalid_cap()	130
12.12.3.3 l4_is_valid_cap()	131

12.13 Capability allocator	132
12.13.1 Detailed Description	132
12.13.2 Function Documentation	132
12.13.2.1 l4re_util_cap_last()	132
12.14 Chunks	133
12.14.1 Detailed Description	133
12.14.2 Function Documentation	133
12.14.2.1 l4shmc_add_chunk()	133
12.14.2.2 l4shmc_chunk_capacity()	134
12.14.2.3 l4shmc_chunk_ptr()	134
12.14.2.4 l4shmc_chunk_signal()	135
12.14.2.5 l4shmc_get_chunk()	135
12.14.2.6 l4shmc_get_chunk_to()	136
12.14.2.7 l4shmc_iterate_chunk()	136
12.15 Comfortable Command Line Parsing	137
12.15.1 Detailed Description	137
12.15.2 Function Documentation	137
12.15.2.1 parse_cmdline()	138
12.16 Console API	140
12.16.1 Detailed Description	140
12.17 Consumer	141
12.17.1 Detailed Description	141
12.17.2 Function Documentation	141
12.17.2.1 l4shmc_chunk_consumed()	141
12.17.2.2 l4shmc_chunk_size()	142
12.17.2.3 l4shmc_enable_chunk()	142
12.17.2.4 l4shmc_is_chunk_ready()	143
12.17.2.5 l4shmc_wait_chunk()	143
12.17.2.6 l4shmc_wait_chunk_to()	143
12.17.2.7 l4shmc_wait_chunk_try()	144

12.18Consumer	145
12.18.1 Detailed Description	145
12.18.2 Function Documentation	145
12.18.2.1 l4shmc_enable_signal()	145
12.18.2.2 l4shmc_wait_any()	146
12.18.2.3 l4shmc_wait_any_to()	146
12.18.2.4 l4shmc_wait_any_try()	147
12.18.2.5 l4shmc_wait_signal()	147
12.18.2.6 l4shmc_wait_signal_to()	147
12.18.2.7 l4shmc_wait_signal_try()	148
12.19DMA API for L4Re	149
12.20DMA Space Interface	150
12.20.1 Detailed Description	150
12.20.2 Typedef Documentation	150
12.20.2.1 l4re_dma_space_t	151
12.20.3 Function Documentation	151
12.20.3.1 l4re_dma_space_associate()	151
12.20.3.2 l4re_dma_space_disassociate()	152
12.20.3.3 l4re_dma_space_map()	152
12.20.3.4 l4re_dma_space_unmap()	153
12.21Dataspace interface	154
12.21.1 Detailed Description	155
12.21.2 Enumeration Type Documentation	155
12.21.2.1 l4re_ds_map_flags	155
12.21.3 Function Documentation	155
12.21.3.1 l4re_ds_allocate()	155
12.21.3.2 l4re_ds_clear()	156
12.21.3.3 l4re_ds_copy_in()	156
12.21.3.4 l4re_ds_flags()	156
12.21.3.5 l4re_ds_info()	157

12.21.3.6 l4re_ds_phys()	157
12.21.3.7 l4re_ds_size()	157
12.22 Debug interface	159
12.22.1 Detailed Description	159
12.22.2 Function Documentation	159
12.22.2.1 l4re_debug_obj_debug()	159
12.23 Debugging API	160
12.23.1 Detailed Description	160
12.24 EDID parsing functionality	161
12.24.1 Detailed Description	161
12.24.2 Enumeration Type Documentation	161
12.24.2.1 Libedid_consts	161
12.24.3 Function Documentation	162
12.24.3.1 libedid_check_header()	162
12.24.3.2 libedid_checksum()	162
12.24.3.3 libedid_dump()	162
12.24.3.4 libedid_dump_standard_timings()	163
12.24.3.5 libedid_num_ext_blocks()	163
12.24.3.6 libedid_pnp_id()	163
12.24.3.7 libedid_prefered_resolution()	164
12.24.3.8 libedid_revision()	164
12.24.3.9 libedid_version()	164
12.25 ELF binary format	166
12.25.1 Detailed Description	178
12.25.2 Macro Definition Documentation	178
12.25.2.1 DF_1_CONFALT	178
12.25.2.2 DF_1_DIRECT	179
12.25.2.3 DF_1_DISPRELDNE	179
12.25.2.4 DF_1_DISPRELPND	179
12.25.2.5 DF_1_ENDFILTEE	179

12.25.2.6 DF_1_GLOBAL	179
12.25.2.7 DF_1_GROUP	180
12.25.2.8 DF_1_INTERPOSE	180
12.25.2.9 DF_1_LOADFLTR	180
12.25.2.10 DF_1_NODEFLIB	180
12.25.2.11 DF_1_NODELETE	180
12.25.2.12 DF_1_NODUMP	181
12.25.2.13 DF_1_NOOPEN	181
12.25.2.14 DF_1_NOW	181
12.25.2.15 DF_1_ORIGIN	181
12.25.2.16 DF_P1_GROUPPERM	181
12.25.2.17 DF_P1_LAZYLOAD	182
12.25.2.18 DT_HIPROC	182
12.25.2.19 DT_LOPROC	182
12.25.2.20 DT_NULL	182
12.25.2.21 EI_CLASS [1/2]	182
12.25.2.22 EI_CLASS [2/2]	183
12.25.2.23 EI_DATA [1/2]	183
12.25.2.24 EI_DATA [2/2]	183
12.25.2.25 EI_OSABI [1/2]	183
12.25.2.26 EI_OSABI [2/2]	184
12.25.2.27 EI_PAD [1/2]	184
12.25.2.28 EI_PAD [2/2]	184
12.25.2.29 EI_VERSION [1/2]	184
12.25.2.30 EI_VERSION [2/2]	185
12.25.2.31 ELFCLASSNONE [1/2]	185
12.25.2.32 ELFCLASSNONE [2/2]	185
12.25.2.33 ELFDATA2LSB [1/2]	185
12.25.2.34 ELFDATA2LSB [2/2]	186
12.25.2.35 ELFDATA2MSB [1/2]	186

12.25.2.36	ELFDATA2MSB [2/2]	186
12.25.2.37	ELFDATANONE [1/2]	186
12.25.2.38	ELFDATANONE [2/2]	187
12.25.2.39	ELFOSABI_AIX	187
12.25.2.40	ELFOSABI_FREEBSD	187
12.25.2.41	ELFOSABI_HPUX [1/2]	187
12.25.2.42	ELFOSABI_HPUX [2/2]	187
12.25.2.43	ELFOSABI_IRIX	188
12.25.2.44	ELFOSABI_LINUX	188
12.25.2.45	ELFOSABI_MODESTO	188
12.25.2.46	ELFOSABI_NETBSD	188
12.25.2.47	ELFOSABI_OPENBSD	188
12.25.2.48	ELFOSABI_SOLARIS	189
12.25.2.49	ELFOSABI_SYSV [1/2]	189
12.25.2.50	ELFOSABI_SYSV [2/2]	189
12.25.2.51	ELFOSABI_TRU64	189
12.25.2.52	EM_ARC	189
12.25.2.53	NT_VERSION	190
12.25.2.54	PT_GNU_EH_FRAME	190
12.25.2.55	PT_GNU_RELRO	190
12.25.2.56	PT_GNU_STACK	190
12.25.2.57	PT_HIOS	190
12.25.2.58	PT_HIPROC	191
12.25.2.59	PT_L4_AUX	191
12.25.2.60	PT_L4_KIP	191
12.25.2.61	PT_L4_STACK	191
12.25.2.62	PT_LOOS	191
12.25.2.63	PT_LOPROC	192
12.25.2.64	SHF_GROUP	192
12.25.2.65	SHF_MASKOS	192

12.25.2.66SHF_TLS	192
12.25.2.67SHT_NUM	192
12.26Error Handling	193
12.26.1 Detailed Description	193
12.26.2 Enumeration Type Documentation	194
12.26.2.1 l4_ipc_tcr_error_t	194
12.26.3 Function Documentation	194
12.26.3.1 l4_error()	194
12.26.3.2 l4_ipc_error()	196
12.26.3.3 l4_ipc_error_code()	197
12.26.3.4 l4_ipc_is_rcv_error()	198
12.26.3.5 l4_ipc_is_snd_error()	198
12.27Error codes	200
12.27.1 Detailed Description	200
12.27.2 Enumeration Type Documentation	200
12.27.2.1 l4_error_code_t	200
12.28Event API	202
12.28.1 Detailed Description	202
12.29Event interface	203
12.29.1 Detailed Description	203
12.29.2 Function Documentation	203
12.29.2.1 l4re_event_get_axis_info()	203
12.29.2.2 l4re_event_get_buffer()	204
12.29.2.3 l4re_event_get_num_streams()	204
12.29.2.4 l4re_event_get_stream_info()	205
12.29.2.5 l4re_event_get_stream_info_for_id()	205
12.30Exception registers	207
12.30.1 Detailed Description	207
12.30.2 Function Documentation	207
12.30.2.1 l4_utcb_exc()	207

12.30.2.2 l4_utcb_exc_is_ex_regs_exception()	208
12.30.2.3 l4_utcb_exc_is_pf()	208
12.30.2.4 l4_utcb_exc_pc()	208
12.30.2.5 l4_utcb_exc_pc_set()	209
12.31 Extended vCPU support	210
12.31.1 Detailed Description	210
12.31.2 Function Documentation	210
12.31.2.1 l4vcpu_ext_alloc()	210
12.32 Factory	212
12.32.1 Detailed Description	213
12.32.2 Function Documentation	213
12.32.2.1 l4_factory_create_factory()	213
12.32.2.2 l4_factory_create_factory_u()	214
12.32.2.3 l4_factory_create_gate()	215
12.32.2.4 l4_factory_create_gate_u()	216
12.32.2.5 l4_factory_create_irq()	217
12.32.2.6 l4_factory_create_irq_u()	218
12.32.2.7 l4_factory_create_task()	219
12.32.2.8 l4_factory_create_task_u()	220
12.32.2.9 l4_factory_create_thread()	221
12.32.2.10 l4_factory_create_thread_u()	222
12.32.2.11 l4_factory_create_vm()	223
12.32.2.12 l4_factory_create_vm_u()	224
12.33 Fiasco extensions	225
12.33.1 Detailed Description	226
12.33.2 Enumeration Type Documentation	226
12.33.2.1 anonymous enum	226
12.33.3 Function Documentation	227
12.33.3.1 fiasco_gdt_get_entry_offset()	227
12.33.3.2 fiasco_gdt_set()	227

12.33.3.3 fiasco_ldt_set()	228
12.33.3.4 fiasco_tbuf_get_status()	228
12.33.3.5 fiasco_tbuf_get_status_phys()	228
12.33.3.6 fiasco_tbuf_log()	229
12.33.3.7 fiasco_tbuf_log_3val()	229
12.33.3.8 fiasco_tbuf_log_binary()	230
12.34 Fiasco real time scheduling extensions	231
12.35 Fiasco-UX Virtual devices	232
12.35.1 Detailed Description	232
12.35.2 Enumeration Type Documentation	232
12.35.2.1 l4_vhw_entry_type	232
12.36 Flex pages	234
12.36.1 Detailed Description	236
12.36.2 Enumeration Type Documentation	236
12.36.2.1 anonymous enum	236
12.36.2.2 anonymous enum	236
12.36.2.3 L4_cap_fpage_rights	237
12.36.2.4 l4_fpage_cacheability_opt_t	238
12.36.2.5 l4_fpage_consts	238
12.36.2.6 L4_fpage_rights	239
12.36.2.7 L4_obj_fpage_ctl	239
12.36.3 Function Documentation	240
12.36.3.1 l4_fpage()	240
12.36.3.2 l4_fpage_all()	240
12.36.3.3 l4_fpage_contains()	241
12.36.3.4 l4_fpage_invalid()	242
12.36.3.5 l4_fpage_ioport()	242
12.36.3.6 l4_fpage_max_order()	242
12.36.3.7 l4_fpage_memaddr()	243
12.36.3.8 l4_fpage_obj()	244

12.36.3.9 l4_fpage_page()	245
12.36.3.10 l4_fpage_rights()	245
12.36.3.11 l4_fpage_set_rights()	246
12.36.3.12 l4_fpage_size()	246
12.36.3.13 l4_fpage_type()	247
12.36.3.14 l4_iofpage()	247
12.36.3.15 l4_is_fpage_writable()	248
12.36.3.16 l4_obj_fpage()	249
12.37 Functions to manipulate the local IDT	250
12.37.1 Detailed Description	250
12.38 IA32 Port I/O API	251
12.38.1 Detailed Description	251
12.38.2 Function Documentation	251
12.38.2.1 l4util_in16()	252
12.38.2.2 l4util_in32()	253
12.38.2.3 l4util_in8()	253
12.38.2.4 l4util_ins16()	254
12.38.2.5 l4util_ins32()	254
12.38.2.6 l4util_ins8()	255
12.38.2.7 l4util_out16()	255
12.38.2.8 l4util_out32()	255
12.38.2.9 l4util_out8()	256
12.38.2.10 l4util_outs16()	256
12.38.2.11 l4util_outs32()	257
12.38.2.12 l4util_outs8()	257
12.39 IO interface	258
12.39.1 Detailed Description	259
12.39.2 Typedef Documentation	259
12.39.2.1 l4io_resource_t	259
12.39.3 Enumeration Type Documentation	259

12.39.3.1 l4io_device_types_t	259
12.39.3.2 l4io_iomem_flags_t	259
12.39.3.3 l4io_resource_types_t	260
12.39.4 Function Documentation	260
12.39.4.1 l4io_has_resource()	260
12.39.4.2 l4io_lookup_device()	261
12.39.4.3 l4io_lookup_resource()	261
12.39.4.4 l4io_release_iomem()	262
12.39.4.5 l4io_release_ioport()	262
12.39.4.6 l4io_request_iomem()	263
12.39.4.7 l4io_request_iomem_region()	263
12.39.4.8 l4io_request_ioport()	264
12.39.4.9 l4io_request_resource_iomem()	264
12.39.4.10 l4io_search_iomem_region()	265
12.40 IPC-Gate API	266
12.40.1 Detailed Description	266
12.40.2 Function Documentation	267
12.40.2.1 l4_ipc_gate_bind_thread()	267
12.40.2.2 l4_ipc_gate_get_infos()	267
12.40.2.3 l4_rcv_ep_bind_thread()	268
12.41 IRQ handling library	269
12.41.1 Detailed Description	269
12.42 IRQs	270
12.42.1 Detailed Description	271
12.42.2 Enumeration Type Documentation	271
12.42.2.1 L4_irq_mode	271
12.42.3 Function Documentation	272
12.42.3.1 l4_irq_attach()	272
12.42.3.2 l4_irq_attach_u()	273
12.42.3.3 l4_irq_detach()	273

12.42.3.4	<code>l4_irq_detach_u()</code>	274
12.42.3.5	<code>l4_irq_mux_chain()</code>	275
12.42.3.6	<code>l4_irq_mux_chain_u()</code>	276
12.42.3.7	<code>l4_irq_receive()</code>	277
12.42.3.8	<code>l4_irq_receive_u()</code>	278
12.42.3.9	<code>l4_irq_trigger()</code>	279
12.42.3.10	<code>l4_irq_trigger_u()</code>	280
12.42.3.11	<code>l4_irq_unmask()</code>	281
12.42.3.12	<code>l4_irq_unmask_u()</code>	282
12.42.3.13	<code>l4_irq_wait()</code>	282
12.42.3.14	<code>l4_irq_wait_u()</code>	283
12.43	Initial Environment	285
12.43.1	Detailed Description	285
12.43.2	Function Documentation	286
12.43.2.1	<code>l4re_env()</code>	286
12.43.2.2	<code>l4re_env_get_cap()</code>	286
12.43.2.3	<code>l4re_env_get_cap_e()</code>	287
12.43.2.4	<code>l4re_env_get_cap_l()</code>	288
12.43.2.5	<code>l4re_kip()</code>	289
12.44	Integer Types	290
12.44.1	Detailed Description	291
12.44.2	Typedef Documentation	291
12.44.2.1	<code>l4_int16_t</code>	291
12.44.2.2	<code>l4_int32_t</code>	292
12.44.2.3	<code>l4_int64_t</code>	292
12.44.2.4	<code>l4_int8_t</code>	292
12.44.2.5	<code>l4_uint16_t</code>	292
12.44.2.6	<code>l4_uint32_t</code>	292
12.44.2.7	<code>l4_uint64_t</code>	292
12.44.2.8	<code>l4_uint8_t</code>	292

12.45Interface for asynchronous ISR handlers with a given IRQ capability.	293
12.45.1 Detailed Description	293
12.45.2 Function Documentation	293
12.45.2.1 l4irq_request_cap()	293
12.46Interface for asynchronous ISR handlers.	295
12.46.1 Detailed Description	295
12.46.2 Function Documentation	295
12.46.2.1 l4irq_release()	295
12.46.2.2 l4irq_request()	296
12.47Interface using direct functionality.	297
12.47.1 Detailed Description	297
12.47.2 Function Documentation	297
12.47.2.1 l4irq_attach()	297
12.47.2.2 l4irq_attach_ft()	298
12.47.2.3 l4irq_attach_thread()	298
12.47.2.4 l4irq_attach_thread_ft()	299
12.47.2.5 l4irq_detach()	299
12.47.2.6 l4irq_unmask()	300
12.47.2.7 l4irq_unmask_and_wait_any()	300
12.47.2.8 l4irq_wait()	300
12.47.2.9 l4irq_wait_any()	302
12.48Interface using direct functionality.	303
12.48.1 Detailed Description	303
12.48.2 Function Documentation	303
12.48.2.1 l4irq_attach_cap()	303
12.48.2.2 l4irq_attach_cap_ft()	304
12.48.2.3 l4irq_attach_thread_cap()	304
12.48.2.4 l4irq_attach_thread_cap_ft()	304
12.49Internal Helpers	306
12.49.1 Detailed Description	306

12.50 Internal constants	307
12.50.1 Detailed Description	308
12.51 Internal functions	309
12.51.1 Detailed Description	309
12.52 Interrupt controller	310
12.52.1 Detailed Description	311
12.52.2 Typedef Documentation	311
12.52.2.1 l4_icu_info_t	311
12.52.3 Enumeration Type Documentation	312
12.52.3.1 L4_icu_flags	312
12.52.4 Function Documentation	312
12.52.4.1 l4_icu_bind()	312
12.52.4.2 l4_icu_bind_u()	313
12.52.4.3 l4_icu_info()	314
12.52.4.4 l4_icu_info_u()	315
12.52.4.5 l4_icu_mask()	315
12.52.4.6 l4_icu_mask_u()	316
12.52.4.7 l4_icu_msi_info()	317
12.52.4.8 l4_icu_msi_info_u()	318
12.52.4.9 l4_icu_set_mode()	318
12.52.4.10 l4_icu_set_mode_u()	319
12.52.4.11 l4_icu_unbind()	320
12.52.4.12 l4_icu_unbind_u()	321
12.52.4.13 l4_icu_unmask()	322
12.52.4.14 l4_icu_unmask_u()	322
12.53 Kernel Debugger	324
12.53.1 Detailed Description	324
12.53.2 Function Documentation	324
12.53.2.1 l4_debugger_global_id()	324
12.53.2.2 l4_debugger_kobj_to_id()	325

12.53.2.3 l4_debugger_set_object_name()	325
12.54 Kernel Interface Page	327
12.54.1 Detailed Description	328
12.54.2 Function Documentation	328
12.54.2.1 l4_kernel_info_version_offset()	328
12.54.2.2 l4_kip_clock()	329
12.54.2.3 l4_kip_clock_lw()	330
12.54.2.4 l4_kip_version()	331
12.54.2.5 l4_kip_version_string()	331
12.55 Kernel Interface Page API	332
12.55.1 Detailed Description	332
12.55.2 Macro Definition Documentation	332
12.55.2.1 l4util_kip_for_each_feature	332
12.55.3 Function Documentation	333
12.55.3.1 l4util_kip_kernel_abi_version()	333
12.55.3.2 l4util_kip_kernel_has_feature()	333
12.55.3.3 l4util_kip_kernel_is_ux()	333
12.55.3.4 l4util_memdesc_vm_high()	334
12.56 Kernel Objects	335
12.56.1 Detailed Description	336
12.57 Kumem allocator utility	337
12.58 Kumem utilities	338
12.58.1 Detailed Description	338
12.58.2 Function Documentation	338
12.58.2.1 kumem_alloc()	338
12.59 L4 IPC Opcodes	340
12.59.1 Detailed Description	340
12.59.2 Enumeration Type Documentation	340
12.59.2.1 L4_icu_opcode	340
12.59.2.2 L4_ipc_gate_ops	341

12.59.2.3 L4_platform_ctl_ops	342
12.59.2.4 L4_task_ops	342
12.59.2.5 L4_thread_ops	342
12.59.2.6 L4_vcon_ops	343
12.60L4 VIRTIO Block Device	344
12.60.1 Detailed Description	344
12.60.2 Enumeration Type Documentation	344
12.60.2.1 L4virtio_block_operations	344
12.60.2.2 L4virtio_block_status	345
12.61L4 VIRTIO Interface	346
12.61.1 Detailed Description	346
12.62L4 VIRTIO Transport Layer	347
12.62.1 Detailed Description	348
12.62.2 Typedef Documentation	349
12.62.2.1 l4virtio_config_queue_t	349
12.62.3 Enumeration Type Documentation	349
12.62.3.1 L4_virtio_cmd	349
12.62.3.2 L4_virtio_irq_status	349
12.62.3.3 L4_virtio_opcodes	350
12.62.3.4 L4virtio_device_ids	350
12.62.3.5 L4virtio_device_status	351
12.62.3.6 L4virtio_feature_bits	351
12.62.4 Function Documentation	351
12.62.4.1 l4virtio_config_queue()	352
12.62.4.2 l4virtio_config_queues()	352
12.62.4.3 l4virtio_device_config()	353
12.62.4.4 l4virtio_device_config_ds()	353
12.62.4.5 l4virtio_device_notification_irq()	354
12.62.4.6 l4virtio_register_ds()	354
12.62.4.7 l4virtio_register_iface()	355

12.62.4.8 l4virtio_set_status()	356
12.63L4 Vbus functions	357
12.63.1 Detailed Description	358
12.63.2 Enumeration Type Documentation	358
12.63.2.1 l4vbus_dma_domain_assign_flags	358
12.63.3 Function Documentation	358
12.63.3.1 l4vbus_assign_dma_domain()	358
12.63.3.2 l4vbus_get_device()	359
12.63.3.3 l4vbus_get_device_by_hid()	360
12.63.3.4 l4vbus_get_hid()	361
12.63.3.5 l4vbus_get_next_device()	361
12.63.3.6 l4vbus_get_resource()	362
12.63.3.7 l4vbus_is_compatible()	363
12.63.3.8 l4vbus_release_resource()	364
12.63.3.9 l4vbus_request_resource()	364
12.63.3.10 l4vbus_vicu_get_cap()	365
12.64L4 kernel object type information	367
12.64.1 Detailed Description	367
12.64.2 Function Documentation	367
12.64.2.1 kobject_typeid()	367
12.65L4Re C Interface	369
12.65.1 Detailed Description	370
12.65.2 Function Documentation	370
12.65.2.1 l4re_inhibitor_acquire()	370
12.66L4Re C++ Interface	372
12.66.1 Detailed Description	373
12.67L4Re Capability API	374
12.67.1 Detailed Description	375
12.67.2 Function Documentation	375
12.67.2.1 make_auto_cap()	375

12.67.2.2 <code>make_auto_del_cap()</code>	375
12.67.2.3 <code>make_ref_cap()</code>	376
12.67.2.4 <code>make_ref_del_cap()</code>	376
12.67.3 Variable Documentation	376
12.67.3.1 <code>cap_alloc</code>	376
12.68L4Re ELF Auxiliary Information	377
12.68.1 Detailed Description	378
12.68.2 Macro Definition Documentation	378
12.68.2.1 <code>L4RE_ELF_AUX_ELEM</code>	378
12.68.2.2 <code>L4RE_ELF_AUX_ELEM_T</code>	378
12.68.3 Enumeration Type Documentation	379
12.68.3.1 anonymous enum	379
12.69L4Re Protocol identifiers	380
12.69.1 Detailed Description	380
12.70L4Re Util C Interface	381
12.71L4Re Util C++ Interface	382
12.71.1 Detailed Description	382
12.72L4vbus GPIO functions	383
12.72.1 Detailed Description	384
12.72.2 Enumeration Type Documentation	384
12.72.2.1 <code>L4vbus_gpio_generic_func</code>	384
12.72.2.2 <code>L4vbus_gpio_pull_modes</code>	384
12.72.3 Function Documentation	385
12.72.3.1 <code>l4vbus_gpio_config_get()</code>	385
12.72.3.2 <code>l4vbus_gpio_config_pad()</code>	386
12.72.3.3 <code>l4vbus_gpio_config_pull()</code>	386
12.72.3.4 <code>l4vbus_gpio_get()</code>	387
12.72.3.5 <code>l4vbus_gpio_multi_config_pad()</code>	388
12.72.3.6 <code>l4vbus_gpio_multi_get()</code>	388
12.72.3.7 <code>l4vbus_gpio_multi_set()</code>	389

12.72.3.8 l4vbus_gpio_multi_setup()	390
12.72.3.9 l4vbus_gpio_set()	391
12.72.3.10 l4vbus_gpio_setup()	392
12.72.3.11 l4vbus_gpio_to_irq()	392
12.73 L4vbus PCI functions	394
12.73.1 Detailed Description	394
12.73.2 Function Documentation	394
12.73.2.1 l4vbus_pci_cfg_read()	394
12.73.2.2 l4vbus_pci_cfg_write()	395
12.73.2.3 l4vbus_pci_irq_enable()	396
12.73.2.4 l4vbus_pcidev_cfg_read()	397
12.73.2.5 l4vbus_pcidev_cfg_write()	398
12.73.2.6 l4vbus_pcidev_irq_enable()	398
12.74 Log interface	400
12.74.1 Detailed Description	400
12.74.2 Function Documentation	400
12.74.2.1 l4re_log_print()	400
12.74.2.2 l4re_log_print_srv()	401
12.74.2.3 l4re_log_printn()	402
12.74.2.4 l4re_log_printn_srv()	403
12.75 Logging interface	404
12.75.1 Detailed Description	404
12.76 Low-Level Thread Functions	405
12.77 Machine Restarting Function	406
12.77.1 Detailed Description	406
12.78 Memory allocator	407
12.78.1 Detailed Description	407
12.78.2 Enumeration Type Documentation	407
12.78.2.1 l4re_ma_flags	408
12.78.3 Function Documentation	408

12.78.3.1 l4re_ma_alloc()	408
12.78.3.2 l4re_ma_alloc_align()	409
12.78.3.3 l4re_ma_alloc_align_srv()	410
12.78.3.4 l4re_ma_free()	411
12.78.3.5 l4re_ma_free_srv()	412
12.79 Memory descriptors (C version)	413
12.79.1 Detailed Description	414
12.79.2 Typedef Documentation	414
12.79.2.1 l4_kernel_info_mem_desc_t	414
12.79.3 Enumeration Type Documentation	414
12.79.3.1 l4_mem_info_sub_type_t	414
12.79.3.2 l4_mem_type_t	415
12.79.4 Function Documentation	415
12.79.4.1 l4_kernel_info_get_mem_desc_end()	415
12.79.4.2 l4_kernel_info_get_mem_desc_is_virtual()	415
12.79.4.3 l4_kernel_info_get_mem_desc_start()	416
12.79.4.4 l4_kernel_info_get_mem_desc_subtype()	416
12.79.4.5 l4_kernel_info_get_mem_desc_type()	416
12.79.4.6 l4_kernel_info_get_num_mem_descs()	417
12.79.4.7 l4_kernel_info_set_mem_desc()	417
12.80 Memory operations.	418
12.80.1 Detailed Description	418
12.80.2 Enumeration Type Documentation	418
12.80.2.1 L4_mem_op_widths	418
12.80.3 Function Documentation	419
12.80.3.1 l4_mem_read()	419
12.80.3.2 l4_mem_write()	420
12.81 Memory related	421
12.81.1 Detailed Description	422
12.81.2 Macro Definition Documentation	422

12.81.2.1 L4_LOG2_PAGESIZE	422
12.81.2.2 L4_LOG2_SUPERPAGESIZE	422
12.81.2.3 L4_PAGEMASK	423
12.81.2.4 L4_SUPERPAGEMASK	423
12.81.2.5 L4_SUPERPAGESIZE	423
12.81.3 Enumeration Type Documentation	423
12.81.3.1 l4_addr_consts_t	423
12.81.4 Function Documentation	424
12.81.4.1 l4_round_page()	424
12.81.4.2 l4_round_size()	425
12.81.4.3 l4_trunc_page()	426
12.81.4.4 l4_trunc_size()	427
12.82 Message Items	428
12.82.1 Detailed Description	428
12.82.2 Enumeration Type Documentation	428
12.82.2.1 l4_msg_item_consts_t	428
12.82.3 Function Documentation	429
12.82.3.1 l4_map_control()	429
12.82.3.2 l4_map_obj_control()	430
12.83 Message Registers (MRs)	431
12.83.1 Detailed Description	431
12.84 Message Tag	432
12.84.1 Detailed Description	433
12.84.2 Typedef Documentation	433
12.84.2.1 l4_msgtag_t	433
12.84.3 Enumeration Type Documentation	434
12.84.3.1 l4_msgtag_flags	434
12.84.3.2 l4_msgtag_protocol	434
12.84.3.3 L4_platform_ctl_proto	435
12.84.4 Function Documentation	435

12.84.4.1 l4_msgtag()	435
12.84.4.2 l4_msgtag_flags()	437
12.84.4.3 l4_msgtag_has_error()	438
12.84.4.4 l4_msgtag_is_exception()	439
12.84.4.5 l4_msgtag_is_io_page_fault()	440
12.84.4.6 l4_msgtag_is_page_fault()	441
12.84.4.7 l4_msgtag_is_preemption()	442
12.84.4.8 l4_msgtag_is_sigma0()	443
12.84.4.9 l4_msgtag_is_sys_exception()	444
12.84.4.10 l4_msgtag_items()	445
12.84.4.11 l4_msgtag_label()	446
12.84.4.12 l4_msgtag_words()	447
12.85 Name-space API	448
12.85.1 Detailed Description	448
12.86 Namespace interface	449
12.86.1 Detailed Description	449
12.86.2 Enumeration Type Documentation	449
12.86.2.1 l4re_ns_register_flags	449
12.86.3 Function Documentation	450
12.86.3.1 l4re_ns_query_srv()	450
12.86.3.2 l4re_ns_query_to_srv()	450
12.86.3.3 l4re_ns_register_obj_srv()	452
12.87 Object Invocation	454
12.87.1 Detailed Description	455
12.87.2 Enumeration Type Documentation	456
12.87.2.1 l4_syscall_flags_t	456
12.87.3 Function Documentation	456
12.87.3.1 l4_ipc()	456
12.87.3.2 l4_ipc_call()	458
12.87.3.3 l4_ipc_receive()	459

12.87.3.4 l4_ipc_reply_and_wait()	460
12.87.3.5 l4_ipc_send()	461
12.87.3.6 l4_ipc_send_and_wait()	463
12.87.3.7 l4_ipc_sleep()	464
12.87.3.8 l4_ipc_wait()	465
12.87.3.9 l4_sndpage_add()	466
12.88 Parent API	467
12.88.1 Detailed Description	467
12.89 Platform Control C API	468
12.89.1 Detailed Description	468
12.89.2 Function Documentation	468
12.89.2.1 l4_platform_ctl_cpu_disable()	468
12.89.2.2 l4_platform_ctl_cpu_enable()	469
12.89.2.3 l4_platform_ctl_system_shutdown()	469
12.89.2.4 l4_platform_ctl_system_suspend()	470
12.90 Producer	471
12.90.1 Detailed Description	471
12.90.2 Function Documentation	471
12.90.2.1 l4shmc_trigger()	471
12.91 Producer	472
12.91.1 Detailed Description	472
12.91.2 Function Documentation	472
12.91.2.1 l4shmc_chunk_ready()	472
12.91.2.2 l4shmc_chunk_ready_sig()	473
12.91.2.3 l4shmc_chunk_try_to_take()	473
12.91.2.4 l4shmc_is_chunk_clear()	474
12.92 Random number support	475
12.92.1 Detailed Description	475
12.92.2 Function Documentation	475
12.92.2.1 l4util_rand()	475

12.92.2.2 l4util_srand()	475
12.93 Realtime API	477
12.94 Region map API	478
12.94.1 Detailed Description	478
12.95 Region map interface	479
12.95.1 Detailed Description	480
12.95.2 Enumeration Type Documentation	480
12.95.2.1 l4re_rm_flags_t	480
12.95.3 Function Documentation	481
12.95.3.1 l4re_rm_attach()	481
12.95.3.2 l4re_rm_attach_srv()	482
12.95.3.3 l4re_rm_detach()	483
12.95.3.4 l4re_rm_detach_ds()	483
12.95.3.5 l4re_rm_detach_ds_unmap()	484
12.95.3.6 l4re_rm_detach_srv()	485
12.95.3.7 l4re_rm_detach_unmap()	486
12.95.3.8 l4re_rm_find()	487
12.95.3.9 l4re_rm_find_srv()	488
12.95.3.10 l4re_rm_free_area()	488
12.95.3.11 l4re_rm_free_area_srv()	489
12.95.3.12 l4re_rm_reserve_area()	489
12.95.3.13 l4re_rm_reserve_area_srv()	490
12.95.3.14 l4re_rm_show_lists()	490
12.96 Scheduler	491
12.96.1 Detailed Description	492
12.96.2 Enumeration Type Documentation	492
12.96.2.1 L4_scheduler_ops	492
12.96.3 Function Documentation	492
12.96.3.1 l4_sched_cpu_set()	492
12.96.3.2 l4_scheduler_idle_time()	493

12.96.3.3 <code>l4_scheduler_info()</code>	494
12.96.3.4 <code>l4_scheduler_is_online()</code>	495
12.96.3.5 <code>l4_scheduler_run_thread()</code>	495
12.97 Server-Side IPC framework	497
12.97.1 Detailed Description	498
12.97.2 Enumeration Type Documentation	498
12.97.2.1 <code>Reply_mode</code>	498
12.98 Shared Memory Library	499
12.98.1 Detailed Description	500
12.98.2 Function Documentation	500
12.98.2.1 <code>l4shmc_area_overhead()</code>	500
12.98.2.2 <code>l4shmc_area_size()</code>	500
12.98.2.3 <code>l4shmc_area_size_free()</code>	501
12.98.2.4 <code>l4shmc_attach()</code>	501
12.98.2.5 <code>l4shmc_attach_to()</code>	502
12.98.2.6 <code>l4shmc_chunk_overhead()</code>	502
12.98.2.7 <code>l4shmc_connect_chunk_signal()</code>	503
12.98.2.8 <code>l4shmc_create()</code>	504
12.99 Sigma0 API	505
12.99.1 Detailed Description	506
12.99.2 Enumeration Type Documentation	506
12.99.2.1 <code>l4sigma0_return_flags_t</code>	506
12.99.3 Function Documentation	506
12.99.3.1 <code>l4sigma0_debug_dump()</code>	506
12.99.3.2 <code>l4sigma0_map_anypage()</code>	507
12.99.3.3 <code>l4sigma0_map_errstr()</code>	507
12.99.3.4 <code>l4sigma0_map_iomem()</code>	508
12.99.3.5 <code>l4sigma0_map_kip()</code>	508
12.99.3.6 <code>l4sigma0_map_mem()</code>	509
12.99.3.7 <code>l4sigma0_map_tbuf()</code>	509

12.99.3.8	l4sigma0_new_client()	510
12.100	Signals	511
12.100.1	Detailed Description	511
12.100.2	Function Documentation	511
12.100.2.1	l4shmc_add_signal()	511
12.100.2.2	l4shmc_attach_signal()	512
12.100.2.3	l4shmc_attach_signal_to()	512
12.100.2.4	l4shmc_check_magic()	513
12.100.2.5	l4shmc_get_signal_to()	513
12.100.2.6	l4shmc_signal_cap()	514
12.101	Small C++ Template Library	515
12.101.1	Detailed Description	516
12.101.2	Function Documentation	516
12.101.2.1	l4max()	516
12.101.2.2	l4min()	517
12.101.2.3	operator new()	518
12.102	Task	519
12.102.1	Detailed Description	520
12.102.2	Enumeration Type Documentation	520
12.102.2.1	l4_unmap_flags_t	520
12.102.3	Function Documentation	521
12.102.3.1	l4_task_add_ku_mem()	521
12.102.3.2	l4_task_cap_equal()	521
12.102.3.3	l4_task_cap_has_child()	522
12.102.3.4	l4_task_cap_valid()	522
12.102.3.5	l4_task_delete_obj()	523
12.102.3.6	l4_task_map()	523
12.102.3.7	l4_task_release_cap()	524
12.102.3.8	l4_task_unmap()	524
12.102.3.9	l4_task_unmap_batch()	525

12.103 Thread	527
12.103.1 Detailed Description	528
12.103.2 Enumeration Type Documentation	529
12.103.2.1 L4_thread_control_flags	529
12.103.2.2 L4_thread_control_mr_indices	529
12.103.2.3 L4_thread_ex_regs_flags	531
12.103.3 Function Documentation	531
12.103.3.1 L4_thread_arm_set_tpidruro()	531
12.103.3.2 L4_thread_ex_regs()	532
12.103.3.3 L4_thread_ex_regs_ret()	533
12.103.3.4 L4_thread_ex_regs_ret_u()	534
12.103.3.5 L4_thread_ex_regs_u()	535
12.103.3.6 L4_thread_modify_sender_add()	536
12.103.3.7 L4_thread_modify_sender_commit()	537
12.103.3.8 L4_thread_modify_sender_start()	537
12.103.3.9 L4_thread_register_del_irq()	537
12.103.3.10 L4_thread_stats_time()	538
12.103.3.11 L4_thread_switch()	538
12.103.3.12 L4_thread_vcpu_control()	539
12.103.3.13 L4_thread_vcpu_control_ext()	539
12.103.3.14 L4_thread_vcpu_control_ext_u()	540
12.103.3.15 L4_thread_vcpu_control_u()	541
12.103.3.16 L4_thread_vcpu_resume_commit()	542
12.103.3.17 L4_thread_vcpu_resume_start()	543
12.103.3.18 L4_thread_yield()	543
12.104 Thread Control Registers (TCRs)	544
12.104.1 Detailed Description	544
12.105 Thread control	545
12.105.1 Detailed Description	545
12.105.2 Function Documentation	546

12.105.2.14_thread_control_alien()	546
12.105.2.24_thread_control_bind()	546
12.105.2.34_thread_control_commit()	547
12.105.2.44_thread_control_exc_handler()	547
12.105.2.54_thread_control_pager()	548
12.105.2.64_thread_control_start()	548
12.105.2.74_thread_control_ux_host_syscall()	549
12.106Timeouts	550
12.106.1Detailed Description	551
12.106.2Macro Definition Documentation	551
12.106.2.1L4_IPC_TIMEOUT_0	551
12.106.3Typedef Documentation	552
12.106.3.14_timeout_s	552
12.106.3.24_timeout_t	552
12.106.4Enumeration Type Documentation	552
12.106.4.14_timeout_abs_validity	552
12.106.5Function Documentation	552
12.106.5.14_ipc_timeout()	552
12.106.5.24_rcv_timeout()	553
12.106.5.34_snd_timeout()	553
12.106.5.44_timeout()	554
12.106.5.54_timeout_abs()	554
12.106.5.64_timeout_get()	555
12.106.5.74_timeout_is_absolute()	556
12.106.5.84_timeout_rel()	557
12.106.5.94_timeout_rel_get()	557
12.106.5.100_utcb_mr64_idx()	558
12.107Timestamp Counter	559
12.107.1Detailed Description	560
12.107.2Function Documentation	560

12.107.2.114_busy_wait_ns()	560
12.107.2.24_busy_wait_us()	561
12.107.2.34_calibrate_tsc()	562
12.107.2.44_get_hz()	562
12.107.2.54_ns_to_tsc()	562
12.107.2.64_rdpmc()	563
12.107.2.74_rdpmc_32()	564
12.107.2.84_rdtsc()	564
12.107.2.94_rdtsc_32()	565
12.107.2.10_tsc_init()	565
12.107.2.11_tsc_to_ns()	566
12.107.2.12_tsc_to_s_and_ns()	566
12.107.2.13_tsc_to_us()	567
12.108Utility Functions	568
12.108.1Detailed Description	569
12.108.2Function Documentation	569
12.108.2.14_sleep()	569
12.108.2.24_touch_ro()	570
12.108.2.34_touch_rw()	571
12.108.2.44_usleep()	571
12.108.2.54util_micros214to()	572
12.108.2.64util_splitlog2_hdl()	572
12.108.2.74util_splitlog2_size()	573
12.109VM API for SVM	575
12.109.1Detailed Description	575
12.110VM API for TZ	576
12.110.1Detailed Description	576
12.111VM API for VMX	577
12.111.1Detailed Description	578
12.111.2Enumeration Type Documentation	578

12.111.2.1anonymous enum	578
12.111.2.24_vm_vmx_caps_regs	579
12.111.2.34_vm_vmx_dfl1_regs	580
12.111.3Function Documentation	580
12.111.3.14_vm_vmx_clear()	580
12.111.3.24_vm_vmx_field_len()	581
12.111.3.34_vm_vmx_field_order()	581
12.111.3.44_vm_vmx_get_caps()	582
12.111.3.54_vm_vmx_get_caps_default1()	582
12.111.3.64_vm_vmx_get_cr2_index()	583
12.111.3.74_vm_vmx_ptr_load()	583
12.111.3.84_vm_vmx_read()	584
12.111.3.94_vm_vmx_read_16()	585
12.111.3.10_vm_vmx_read_32()	586
12.111.3.114_vm_vmx_read_64()	586
12.111.3.12_vm_vmx_read_nat()	587
12.111.3.13_vm_vmx_write()	588
12.111.3.14_vm_vmx_write_16()	588
12.111.3.15_vm_vmx_write_32()	589
12.111.3.16_vm_vmx_write_64()	590
12.111.3.17_vm_vmx_write_nat()	590
12.112bus API	593
12.112.1Detailed Description	593
12.113Video API	594
12.113.1Detailed Description	595
12.113.2Typedef Documentation	595
12.113.2.14re_video_view_t	595
12.113.3Enumeration Type Documentation	596
12.113.3.14re_video_goos_info_flags_t	596
12.113.3.24re_video_view_info_flags_t	596

12.113.4Function Documentation	596
12.113.4.14re_video_goos_create_buffer()	597
12.113.4.24re_video_goos_create_view()	597
12.113.4.34re_video_goos_delete_buffer()	597
12.113.4.44re_video_goos_delete_view()	599
12.113.4.54re_video_goos_get_static_buffer()	599
12.113.4.64re_video_goos_get_view()	599
12.113.4.74re_video_goos_info()	600
12.113.4.84re_video_goos_refresh()	600
12.113.4.94re_video_view_get_info()	601
12.113.4.104re_video_view_refresh()	601
12.113.4.114re_video_view_set_info()	602
12.113.4.124re_video_view_set_viewport()	602
12.113.4.134re_video_view_stack()	602
12.114Virtual Console	604
12.114.1Detailed Description	605
12.114.2Enumeration Type Documentation	605
12.114.2.1L4_vcon_i_flags	605
12.114.2.2L4_vcon_l_flags	606
12.114.2.3L4_vcon_o_flags	606
12.114.2.4L4_vcon_size_consts	606
12.114.3Function Documentation	607
12.114.3.1L4_vcon_get_attr()	607
12.114.3.2L4_vcon_get_attr_u()	608
12.114.3.3L4_vcon_read()	608
12.114.3.4L4_vcon_read_u()	609
12.114.3.5L4_vcon_read_with_flags()	610
12.114.3.6L4_vcon_send()	611
12.114.3.7L4_vcon_send_u()	612
12.114.3.8L4_vcon_set_attr()	613

12.114.3.94_vcon_set_attr_u()	614
12.114.3.10_vcon_write()	614
12.114.3.11_vcon_write_u()	615
12.115 Virtual Machines	617
12.115.1 Detailed Description	617
12.116 Virtual Registers (UTCBs)	618
12.116.1 Detailed Description	619
12.116.2 Typedef Documentation	619
12.116.2.1 utcb_t	619
12.116.3 Function Documentation	620
12.116.3.1 utcb_br()	620
12.116.3.2 utcb_mr()	620
12.116.3.3 utcb_tcr()	621
12.117 amd64 Virtual Registers (UTCB)	622
12.117.1 Detailed Description	622
12.118 CPU API	623
12.118.1 Detailed Description	624
12.118.2 Enumeration Type Documentation	624
12.118.2.1 vcpu_state_flags	624
12.118.2.2 vcpu_state_offset	624
12.118.2.3 vcpu_sticky_flags	625
12.119 CPU Support Library	626
12.119.1 Detailed Description	627
12.119.2 Function Documentation	627
12.119.2.1 vcpu_irq_disable()	627
12.119.2.2 vcpu_irq_disable_save()	628
12.119.2.3 vcpu_irq_enable()	628
12.119.2.4 vcpu_irq_restore()	629
12.119.2.5 vcpu_is_irq_entry()	630
12.119.2.6 vcpu_is_page_fault_entry()	631
12.119.2.7 vcpu_print_state()	632
12.119.2.8 vcpu_wait_for_event()	632
12.120 x86 Virtual Registers (UTCB)	634
12.120.1 Detailed Description	634
12.120.2 Enumeration Type Documentation	634
12.120.2.1 utcb_consts_x86	634

13 Namespace Documentation	637
13.1 cxx Namespace Reference	637
13.1.1 Detailed Description	639
13.1.2 Typedef Documentation	639
13.1.2.1 H_list_item	639
13.2 cxx::Bits Namespace Reference	639
13.2.1 Detailed Description	640
13.3 L4 Namespace Reference	640
13.3.1 Detailed Description	644
13.3.2 Enumeration Type Documentation	644
13.3.2.1 anonymous enum	644
13.3.3 Function Documentation	644
13.3.3.1 cap_cast() [1/2]	644
13.3.3.2 cap_cast() [2/2]	645
13.3.3.3 cap_dynamic_cast()	646
13.3.3.4 cap_reinterpret_cast() [1/2]	647
13.3.3.5 cap_reinterpret_cast() [2/2]	648
13.3.3.6 throw_ipc_exception() [1/2]	649
13.3.3.7 throw_ipc_exception() [2/2]	650
13.4 L4::lpc Namespace Reference	651
13.4.1 Detailed Description	653
13.4.2 Function Documentation	653
13.4.2.1 buf_cp_in()	654
13.4.2.2 buf_cp_out()	654
13.4.2.3 buf_in()	655
13.4.2.4 make_cap()	655
13.4.2.5 make_cap_full()	656
13.4.2.6 make_cap_rw()	657
13.4.2.7 make_cap_rws()	658
13.4.2.8 msg_ptr()	658

13.4.2.9	<code>read()</code>	659
13.4.2.10	<code>str_cp_in()</code>	659
13.5	<code>L4::lpc::Msg</code> Namespace Reference	660
13.5.1	Detailed Description	661
13.5.2	Enumeration Type Documentation	661
13.5.2.1	anonymous enum	661
13.5.3	Function Documentation	662
13.5.3.1	<code>align_to()</code> [1/2]	662
13.5.3.2	<code>align_to()</code> [2/2]	663
13.5.3.3	<code>check_size()</code> [1/2]	663
13.5.3.4	<code>check_size()</code> [2/2]	664
13.5.3.5	<code>msg_add()</code>	664
13.5.3.6	<code>msg_get()</code>	665
13.6	<code>L4::lpc_svr</code> Namespace Reference	666
13.6.1	Detailed Description	667
13.7	<code>L4::Typeid</code> Namespace Reference	667
13.7.1	Detailed Description	667
13.8	<code>L4::Types</code> Namespace Reference	667
13.8.1	Detailed Description	668
13.9	<code>L4Re</code> Namespace Reference	668
13.9.1	Detailed Description	670
13.9.2	Typedef Documentation	670
13.9.2.1	<code>Shared_cap</code>	670
13.9.2.2	<code>shared_cap</code>	670
13.9.2.3	<code>Shared_del_cap</code>	671
13.9.2.4	<code>shared_del_cap</code>	671
13.9.2.5	<code>Unique_cap</code>	672
13.9.2.6	<code>unique_cap</code>	672
13.9.2.7	<code>Unique_del_cap</code>	673
13.9.2.8	<code>unique_del_cap</code>	673

13.9.3	Function Documentation	674
13.9.3.1	chkcap()	674
13.9.3.2	chkipc()	675
13.9.3.3	chksys() [1/3]	676
13.9.3.4	chksys() [2/3]	677
13.9.3.5	chksys() [3/3]	678
13.9.3.6	make_shared_cap()	679
13.9.3.7	make_shared_del_cap()	680
13.9.3.8	make_unique_cap()	681
13.9.3.9	make_unique_del_cap()	681
13.10	L4Re::Util Namespace Reference	682
13.10.1	Detailed Description	684
13.10.2	Typedef Documentation	684
13.10.2.1	Shared_cap	684
13.10.2.2	shared_cap	685
13.10.2.3	Shared_del_cap	686
13.10.2.4	shared_del_cap	686
13.10.2.5	Unique_cap	687
13.10.2.6	unique_cap	688
13.10.2.7	Unique_del_cap	688
13.10.2.8	unique_del_cap	689
13.10.3	Function Documentation	690
13.10.3.1	make_shared_cap()	690
13.10.3.2	make_shared_del_cap()	690
13.10.3.3	make_unique_cap()	691
13.10.3.4	make_unique_del_cap()	691
13.11	L4Re::Vfs Namespace Reference	692
13.11.1	Detailed Description	693
13.12	L4virtio Namespace Reference	693
13.12.1	Detailed Description	693

14 Data Structure Documentation	695
14.1 <code>cxx::Auto_ptr< T ></code> Class Template Reference	695
14.1.1 Detailed Description	696
14.1.2 Member Typedef Documentation	696
14.1.2.1 <code>Ref_type</code>	696
14.1.3 Constructor & Destructor Documentation	697
14.1.3.1 <code>Auto_ptr()</code> [1/2]	697
14.1.3.2 <code>Auto_ptr()</code> [2/2]	697
14.1.3.3 <code>~Auto_ptr()</code>	697
14.1.4 Member Function Documentation	697
14.1.4.1 <code>get()</code>	698
14.1.4.2 <code>operator Priv_type *()</code>	698
14.1.4.3 <code>operator*()</code>	698
14.1.4.4 <code>operator->()</code>	698
14.1.4.5 <code>operator=()</code>	698
14.1.4.6 <code>release()</code>	699
14.2 <code>cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC ></code> Class Template Reference	700
14.2.1 Detailed Description	702
14.2.2 Constructor & Destructor Documentation	702
14.2.2.1 <code>Avl_map()</code>	702
14.2.3 Member Function Documentation	703
14.2.3.1 <code>insert()</code>	703
14.2.3.2 <code>operator[]()</code> [1/2]	703
14.2.3.3 <code>operator[]()</code> [2/2]	704
14.3 <code>cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC ></code> Class Template Reference	704
14.3.1 Detailed Description	706
14.4 <code>cxx::Avl_tree< Node, Get_key, Compare ></code> Class Template Reference	707
14.4.1 Detailed Description	710
14.4.2 Member Function Documentation	711
14.4.2.1 <code>insert()</code>	711

14.4.2.2	remove()	711
14.5	cxx::Avl_tree_node Class Reference	712
14.5.1	Detailed Description	714
14.6	cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	715
14.6.1	Detailed Description	716
14.6.2	Member Enumeration Documentation	717
14.6.2.1	anonymous enum	717
14.6.3	Member Function Documentation	717
14.6.3.1	free_objects()	717
14.6.3.2	total_objects()	718
14.7	cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	718
14.7.1	Detailed Description	720
14.7.2	Member Enumeration Documentation	721
14.7.2.1	anonymous enum	721
14.7.3	Member Function Documentation	721
14.7.3.1	alloc()	721
14.7.3.2	free()	721
14.7.3.3	free_objects()	722
14.7.3.4	total_objects()	722
14.8	cxx::Bitfield< T, LSB, MSB > Class Template Reference	723
14.8.1	Detailed Description	724
14.8.2	Member Typedef Documentation	724
14.8.2.1	Bits_type	724
14.8.2.2	Ref	725
14.8.2.3	Ref_unshifted	725
14.8.2.4	Shift_type	725
14.8.2.5	Val	725
14.8.2.6	Val_unshifted	725
14.8.3	Member Enumeration Documentation	725
14.8.3.1	anonymous enum	725

14.8.3.2	Masks	726
14.8.4	Member Function Documentation	726
14.8.4.1	get()	726
14.8.4.2	get_unshifted()	727
14.8.4.3	set()	727
14.8.4.4	set_dirty()	728
14.8.4.5	set_unshifted()	729
14.8.4.6	set_unshifted_dirty()	730
14.8.4.7	val()	731
14.8.4.8	val_dirty()	732
14.8.4.9	val_unshifted()	732
14.9	cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference	733
14.9.1	Detailed Description	734
14.10	cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference	734
14.10.1	Detailed Description	735
14.11	cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference	736
14.11.1	Detailed Description	737
14.12	cxx::Bitmap< BITS > Class Template Reference	737
14.12.1	Detailed Description	739
14.12.2	Constructor & Destructor Documentation	740
14.12.2.1	Bitmap()	740
14.12.3	Member Function Documentation	740
14.12.3.1	scan_zero()	740
14.13	cxx::Bitmap_base Class Reference	741
14.13.1	Detailed Description	744
14.13.2	Member Enumeration Documentation	744
14.13.2.1	anonymous enum	744
14.13.3	Member Function Documentation	745
14.13.3.1	bit() [1/2]	745
14.13.3.2	bit() [2/2]	746

14.13.3.3 <code>bit_index()</code>	746
14.13.3.4 <code>chars()</code>	747
14.13.3.5 <code>clear_bit()</code>	747
14.13.3.6 <code>operator[]()</code> [1/2]	748
14.13.3.7 <code>operator[]()</code> [2/2]	749
14.13.3.8 <code>scan_zero()</code>	749
14.13.3.9 <code>set_bit()</code>	750
14.13.3.10 <code>word_index()</code>	751
14.13.3.11 <code>words()</code>	752
14.14 <code>cxx::Bitmap_base::Bit</code> Class Reference	752
14.14.1 Detailed Description	752
14.15 <code>cxx::Bitmap_base::Char< BITS ></code> Class Template Reference	753
14.15.1 Detailed Description	753
14.16 <code>cxx::Bitmap_base::Word< BITS ></code> Class Template Reference	753
14.16.1 Detailed Description	754
14.17 <code>cxx::Bits::Avl_map_get_key< KEY_TYPE ></code> Struct Template Reference	755
14.17.1 Detailed Description	755
14.18 <code>cxx::Bits::Avl_set_get_key< KEY_TYPE ></code> Struct Template Reference	755
14.18.1 Detailed Description	756
14.19 <code>cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY ></code> Class Template Reference	756
14.19.1 Detailed Description	759
14.19.2 Member Enumeration Documentation	760
14.19.2.1 anonymous enum	760
14.19.3 Constructor & Destructor Documentation	760
14.19.3.1 <code>Base_avl_set()</code> [1/2]	760
14.19.3.2 <code>Base_avl_set()</code> [2/2]	761
14.19.4 Member Function Documentation	761
14.19.4.1 <code>begin()</code> [1/2]	761
14.19.4.2 <code>begin()</code> [2/2]	762
14.19.4.3 <code>end()</code> [1/2]	762

14.19.4.4	end() [2/2]	763
14.19.4.5	erase()	763
14.19.4.6	find_node()	763
14.19.4.7	insert()	764
14.19.4.8	lower_bound_node()	764
14.19.4.9	rbegin() [1/2]	765
14.19.4.10	rbegin() [2/2]	765
14.19.4.11	remove()	765
14.19.4.12	end() [1/2]	766
14.19.4.13	end() [2/2]	766
14.20	cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference	767
14.20.1	Detailed Description	767
14.20.2	Member Function Documentation	768
14.20.2.1	operator*()	768
14.20.2.2	operator->()	768
14.20.2.3	valid()	768
14.21	cxx::Bits::Basic_list< POLICY > Class Template Reference	769
14.21.1	Detailed Description	770
14.21.2	Member Function Documentation	770
14.21.2.1	clear()	770
14.21.2.2	iter()	771
14.22	cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	771
14.22.1	Detailed Description	775
14.22.2	Member Function Documentation	775
14.22.2.1	begin() [1/2]	776
14.22.2.2	begin() [2/2]	776
14.22.2.3	dir() [1/2]	776
14.22.2.4	dir() [2/2]	777
14.22.2.5	end() [1/2]	778
14.22.2.6	end() [2/2]	779

14.22.2.7 find()	779
14.22.2.8 find_node()	780
14.22.2.9 lower_bound_node()	780
14.22.2.10 begin() [1/2]	781
14.22.2.11 rbegin() [2/2]	782
14.22.2.12 remove_all()	782
14.22.2.13 remove_tree()	783
14.22.2.14 end() [1/2]	784
14.22.2.15 end() [2/2]	784
14.23 cxx::Bits::Bst_node Class Reference	785
14.23.1 Detailed Description	786
14.24 cxx::Bits::Direction Struct Reference	787
14.24.1 Detailed Description	788
14.24.2 Member Enumeration Documentation	788
14.24.2.1 Direction_e	788
14.24.3 Member Function Documentation	788
14.24.3.1 operator"!"()	788
14.25 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference	789
14.25.1 Detailed Description	790
14.25.2 Member Function Documentation	790
14.25.2.1 pop_front()	790
14.26 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference	791
14.26.1 Detailed Description	791
14.27 cxx::H_list< T, POLICY > Class Template Reference	792
14.27.1 Detailed Description	795
14.27.2 Member Function Documentation	795
14.27.2.1 erase()	795
14.27.2.2 insert()	796
14.27.2.3 insert_after()	796
14.27.2.4 insert_before()	797

14.27.2.5 iter()	797
14.27.2.6 pop_front()	798
14.27.2.7 remove()	798
14.27.2.8 replace()	798
14.28cxx::H_list_item_t< ELEM_TYPE > Class Template Reference	799
14.28.1 Detailed Description	800
14.28.2 Constructor & Destructor Documentation	800
14.28.2.1 H_list_item_t()	800
14.28.2.2 ~H_list_item_t()	801
14.29cxx::H_list_t< T > Struct Template Reference	801
14.29.1 Detailed Description	803
14.30cxx::List< D, Alloc > Class Template Reference	804
14.30.1 Detailed Description	805
14.30.2 Member Function Documentation	805
14.30.2.1 items()	805
14.30.2.2 operator[]() [1/2]	805
14.30.2.3 operator[]() [2/2]	805
14.30.2.4 push_back()	806
14.30.2.5 push_front()	806
14.30.2.6 remove()	806
14.30.2.7 size()	806
14.31cxx::List< D, Alloc >::Iter Class Reference	807
14.31.1 Detailed Description	807
14.32cxx::List_alloc Class Reference	808
14.32.1 Detailed Description	808
14.32.2 Constructor & Destructor Documentation	808
14.32.2.1 List_alloc()	809
14.32.3 Member Function Documentation	809
14.32.3.1 alloc()	809
14.32.3.2 alloc_max()	810

14.32.3.3 <code>avail()</code>	811
14.32.3.4 <code>free()</code>	812
14.33cxx::List_item Class Reference	812
14.33.1 Detailed Description	814
14.33.2 Member Function Documentation	814
14.33.2.1 <code>get_next_item()</code>	814
14.33.2.2 <code>get_prev_item()</code>	814
14.33.2.3 <code>insert_next_item()</code>	815
14.33.2.4 <code>insert_prev_item()</code>	815
14.33.2.5 <code>push_back()</code>	815
14.33.2.6 <code>push_front()</code>	816
14.33.2.7 <code>remove()</code>	816
14.33.2.8 <code>remove_me()</code>	817
14.34cxx::List_item::Iter Class Reference	818
14.34.1 Detailed Description	820
14.34.2 Member Function Documentation	820
14.34.2.1 <code>remove_me()</code>	820
14.35cxx::List_item::T_iter< T, Poly > Class Template Reference	821
14.35.1 Detailed Description	823
14.36cxx::Lt_functor< Obj > Struct Template Reference	824
14.36.1 Detailed Description	824
14.37cxx::New_allocator< _Type > Class Template Reference	824
14.37.1 Detailed Description	825
14.38cxx::Nothrow Class Reference	825
14.38.1 Detailed Description	825
14.39cxx::Pair< First, Second > Struct Template Reference	826
14.39.1 Detailed Description	826
14.39.2 Constructor & Destructor Documentation	827
14.39.2.1 <code>Pair()</code>	827
14.40cxx::Pair_first_compare< Cmp, Typ > Class Template Reference	827

14.40.1 Detailed Description	828
14.40.2 Constructor & Destructor Documentation	828
14.40.2.1 Pair_first_compare()	828
14.40.3 Member Function Documentation	828
14.40.3.1 operator()()	829
14.41 cxx::Ref_obj_list_item< T > Struct Template Reference	829
14.41.1 Detailed Description	830
14.42 cxx::Ref_ptr< T, CNT > Class Template Reference	830
14.42.1 Detailed Description	832
14.42.2 Constructor & Destructor Documentation	833
14.42.2.1 Ref_ptr() [1/3]	833
14.42.2.2 Ref_ptr() [2/3]	834
14.42.2.3 Ref_ptr() [3/3]	834
14.42.3 Member Function Documentation	834
14.42.3.1 get()	834
14.42.3.2 ptr()	835
14.42.3.3 release()	835
14.43 cxx::S_list< T, POLICY > Class Template Reference	836
14.43.1 Detailed Description	839
14.43.2 Member Function Documentation	839
14.43.2.1 pop_front()	839
14.44 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference	839
14.44.1 Detailed Description	842
14.44.2 Member Function Documentation	842
14.44.2.1 alloc()	842
14.44.2.2 free()	842
14.45 cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference	843
14.45.1 Detailed Description	845
14.45.2 Member Function Documentation	846
14.45.2.1 alloc()	846

14.46cxx::static_vector< T, IDX > Class Template Reference	846
14.46.1 Detailed Description	847
14.47cxx::String Class Reference	848
14.47.1 Detailed Description	850
14.47.2 Constructor & Destructor Documentation	851
14.47.2.1 String()	851
14.47.3 Member Function Documentation	851
14.47.3.1 find()	851
14.47.3.2 from_dec()	852
14.47.3.3 from_hex()	853
14.47.3.4 starts_with()	854
14.48cxx::Weak_ref< T > Class Template Reference	855
14.48.1 Detailed Description	856
14.49cxx::Weak_ref_base Class Reference	857
14.49.1 Detailed Description	859
14.50Elf32_Dyn Struct Reference	860
14.50.1 Detailed Description	860
14.50.2 Field Documentation	860
14.50.2.1 d_val	860
14.51Elf32_Ehdr Struct Reference	861
14.51.1 Detailed Description	862
14.51.2 Field Documentation	862
14.51.2.1 e_phnum	862
14.51.2.2 e_shnum	862
14.52Elf32_Phdr Struct Reference	863
14.52.1 Detailed Description	863
14.53Elf32_Shdr Struct Reference	864
14.53.1 Detailed Description	865
14.54Elf32_Sym Struct Reference	865
14.54.1 Detailed Description	866

14.55Elf64_Dyn Struct Reference	866
14.55.1 Detailed Description	866
14.55.2 Field Documentation	867
14.55.2.1 d_val	867
14.56Elf64_Ehdr Struct Reference	867
14.56.1 Detailed Description	868
14.56.2 Field Documentation	868
14.56.2.1 e_phnum	868
14.56.2.2 e_shnum	869
14.57Elf64_Phdr Struct Reference	869
14.57.1 Detailed Description	870
14.58Elf64_Shdr Struct Reference	870
14.58.1 Detailed Description	871
14.59Elf64_Sym Struct Reference	871
14.59.1 Detailed Description	872
14.60L4::Alloc_list Class Reference	872
14.60.1 Detailed Description	873
14.61L4::Arm_smccc Class Reference	873
14.61.1 Detailed Description	873
14.61.2 Member Function Documentation	873
14.61.2.1 call()	874
14.62L4::Base_exception Class Reference	874
14.62.1 Detailed Description	875
14.63L4::Basic_registry Class Reference	876
14.63.1 Detailed Description	877
14.63.2 Member Function Documentation	877
14.63.2.1 dispatch()	877
14.63.2.2 find()	877
14.64L4::Bounds_error Class Reference	878
14.64.1 Detailed Description	880

14.65L4::Cap< T > Class Template Reference	881
14.65.1 Detailed Description	883
14.65.2 Constructor & Destructor Documentation	883
14.65.2.1 Cap() [1/4]	883
14.65.2.2 Cap() [2/4]	884
14.65.2.3 Cap() [3/4]	884
14.65.2.4 Cap() [4/4]	884
14.65.3 Member Function Documentation	885
14.65.3.1 copy()	885
14.65.3.2 move()	885
14.66L4::Cap_base Class Reference	886
14.66.1 Detailed Description	887
14.66.2 Member Enumeration Documentation	888
14.66.2.1 Cap_type	888
14.66.2.2 No_init_type	888
14.66.3 Constructor & Destructor Documentation	888
14.66.3.1 Cap_base() [1/2]	888
14.66.3.2 Cap_base() [2/2]	889
14.66.4 Member Function Documentation	889
14.66.4.1 cap()	889
14.66.4.2 copy()	890
14.66.4.3 fpage()	891
14.66.4.4 is_valid()	892
14.66.4.5 move()	893
14.66.4.6 snd_base()	894
14.66.4.7 validate() [1/2]	895
14.66.4.8 validate() [2/2]	896
14.66.5 Field Documentation	896
14.66.5.1 _c	896
14.67L4::Com_error Class Reference	897

14.67.1 Detailed Description	899
14.67.2 Constructor & Destructor Documentation	899
14.67.2.1 Com_error()	899
14.68L4::Debugger Class Reference	899
14.68.1 Detailed Description	902
14.68.2 Member Function Documentation	902
14.68.2.1 get_object_name()	902
14.68.2.2 global_id()	903
14.68.2.3 kobj_to_id()	903
14.68.2.4 query_log_name()	904
14.68.2.5 query_log_typeid()	904
14.68.2.6 set_object_name()	905
14.68.2.7 switch_log()	905
14.69L4::Element_already_exists Class Reference	906
14.69.1 Detailed Description	907
14.70L4::Element_not_found Class Reference	908
14.70.1 Detailed Description	909
14.71L4::Exception Class Reference	910
14.71.1 Detailed Description	910
14.71.2 Member Typedef Documentation	910
14.71.2.1 RpcS	910
14.72L4::Exception_tracer Class Reference	911
14.72.1 Detailed Description	912
14.73L4::Factory Class Reference	913
14.73.1 Detailed Description	915
14.73.2 Member Function Documentation	916
14.73.2.1 create() [1/2]	916
14.73.2.2 create() [2/2]	917
14.73.2.3 create_factory()	918
14.73.2.4 create_gate()	919

14.73.2.5 create_irq()	920
14.73.2.6 create_task()	921
14.73.2.7 create_thread()	922
14.73.2.8 create_vm()	923
14.74L4::Factory::Lstr Struct Reference	924
14.74.1 Detailed Description	925
14.74.2 Constructor & Destructor Documentation	925
14.74.2.1 Lstr()	925
14.75L4::Factory::Nil Struct Reference	925
14.75.1 Detailed Description	926
14.76L4::Factory::S Class Reference	926
14.76.1 Detailed Description	927
14.76.2 Constructor & Destructor Documentation	927
14.76.2.1 S() [1/2]	927
14.76.2.2 S() [2/2]	927
14.76.3 Member Function Documentation	928
14.76.3.1 operator l4_msgtag_t()	928
14.76.3.2 operator<<() [1/6]	928
14.76.3.3 operator<<() [2/6]	929
14.76.3.4 operator<<() [3/6]	929
14.76.3.5 operator<<() [4/6]	929
14.76.3.6 operator<<() [5/6]	930
14.76.3.7 operator<<() [6/6]	930
14.77L4::lcu Class Reference	931
14.77.1 Detailed Description	933
14.77.2 Member Function Documentation	933
14.77.2.1 bind()	933
14.77.2.2 info()	935
14.77.2.3 mask()	936
14.77.2.4 msi_info()	937

14.77.2.5 <code>set_mode()</code>	938
14.77.2.6 <code>unbind()</code>	938
14.78L4::lcu::Info Class Reference	939
14.78.1 Detailed Description	940
14.79L4::Invalid_capability Class Reference	940
14.79.1 Detailed Description	942
14.79.2 Constructor & Destructor Documentation	943
14.79.2.1 <code>Invalid_capability()</code>	943
14.79.3 Member Function Documentation	943
14.79.3.1 <code>cap()</code>	943
14.80L4::io_pager Class Reference	944
14.80.1 Detailed Description	945
14.80.2 Member Function Documentation	946
14.80.2.1 <code>io_page_fault()</code>	946
14.81L4::lomu Class Reference	946
14.81.1 Detailed Description	948
14.81.2 Member Function Documentation	949
14.81.2.1 <code>bind()</code>	949
14.81.2.2 <code>unbind()</code>	949
14.82L4::IOModifier Class Reference	949
14.82.1 Detailed Description	950
14.83L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference	950
14.83.1 Detailed Description	952
14.84L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference	953
14.84.1 Detailed Description	954
14.85L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference	954
14.85.1 Detailed Description	956
14.86L4::lpc::As_value< T > Struct Template Reference	956
14.86.1 Detailed Description	957
14.87L4::lpc::Buf_item Class Reference	957

14.87.1 Detailed Description	958
14.88L4::lpc::Call Struct Reference	958
14.88.1 Detailed Description	959
14.89L4::lpc::Call_t< RIGHTS > Struct Template Reference	959
14.89.1 Detailed Description	960
14.90L4::lpc::Call_zero_send_timeout Struct Reference	961
14.90.1 Detailed Description	962
14.91L4::lpc::Cap< T > Class Template Reference	962
14.91.1 Detailed Description	964
14.91.2 Member Enumeration Documentation	964
14.91.2.1 anonymous enum	964
14.91.3 Constructor & Destructor Documentation	964
14.91.3.1 Cap()	965
14.91.4 Member Function Documentation	965
14.91.4.1 from_ci()	965
14.92L4::lpc::Gen_fpage< T > Class Template Reference	965
14.92.1 Detailed Description	967
14.92.2 Member Function Documentation	967
14.92.2.1 cap_received()	967
14.92.2.2 id_received()	968
14.92.2.3 is_compound()	968
14.92.2.4 local_id_received()	968
14.93L4::lpc::In_out< T > Struct Template Reference	969
14.93.1 Detailed Description	969
14.94L4::lpc::lostream Class Reference	969
14.94.1 Detailed Description	972
14.94.2 Constructor & Destructor Documentation	972
14.94.2.1 lostream()	972
14.94.3 Member Function Documentation	973
14.94.3.1 call()	973

14.94.3.2 reply_and_wait() [1/2]	974
14.94.3.3 reply_and_wait() [2/2]	974
14.94.3.4 reset()	975
14.95L4::lpc::Istream Class Reference	976
14.95.1 Detailed Description	979
14.95.2 Constructor & Destructor Documentation	979
14.95.2.1 Istream()	980
14.95.3 Member Function Documentation	980
14.95.3.1 get() [1/3]	980
14.95.3.2 get() [2/3]	981
14.95.3.3 get() [3/3]	981
14.95.3.4 receive()	982
14.95.3.5 reset()	983
14.95.3.6 skip()	983
14.95.3.7 tag() [1/2]	984
14.95.3.8 tag() [2/2]	985
14.95.3.9 wait() [1/2]	985
14.95.3.10 wait() [2/2]	985
14.96L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	986
14.96.1 Detailed Description	987
14.97L4::lpc::Msg::Cls_buffer Struct Reference	987
14.97.1 Detailed Description	988
14.98L4::lpc::Msg::Cls_data Struct Reference	989
14.98.1 Detailed Description	989
14.99L4::lpc::Msg::Cls_item Struct Reference	990
14.99.1 Detailed Description	990
14.100L4::lpc::Msg::Dir_in Struct Reference	991
14.100.1 Detailed Description	991
14.101L4::lpc::Msg::Dir_out Struct Reference	992
14.101.1 Detailed Description	992

14.1024::lpc::Msg::Do_in_data Struct Reference	993
14.102.1 Detailed Description	993
14.1084::lpc::Msg::Do_in_items Struct Reference	994
14.103.1 Detailed Description	994
14.1044::lpc::Msg::Do_out_data Struct Reference	995
14.104.1 Detailed Description	996
14.1054::lpc::Msg::Do_out_items Struct Reference	996
14.105.1 Detailed Description	997
14.1064::lpc::Msg::Do_rcv_buffers Struct Reference	997
14.106.1 Detailed Description	998
14.1074::lpc::Msg::Elem< Array< A, LEN > &> Struct Template Reference	999
14.107.1 Detailed Description	999
14.1084::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference	1000
14.108.1 Detailed Description	1000
14.1094::lpc::Msg::Elem< Array_ref< A, LEN > &> Struct Template Reference	1001
14.109.1 Detailed Description	1001
14.1104::lpc::Msg::Is_valid_rpc_type< T > Struct Template Reference	1002
14.110.1 Detailed Description	1003
14.1114::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference	1004
14.111.1 Detailed Description	1004
14.1124::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	1004
14.112.1 Detailed Description	1005
14.1134::lpc::Msg_ptr< T > Class Template Reference	1005
14.113.1 Detailed Description	1006
14.113.2 Constructor & Destructor Documentation	1006
14.113.2.1 Msg_ptr()	1006
14.1144::lpc::Opt< T > Struct Template Reference	1006
14.114.1 Detailed Description	1008
14.1154::lpc::Ostream Class Reference	1008
14.115.1 Detailed Description	1011

14.115.2	Member Function Documentation	1011
14.115.2.1	put() [1/2]	1011
14.115.2.2	put() [2/2]	1012
14.115.2.3	send()	1012
14.115.2.4	tag() [1/2]	1013
14.115.2.5	tag() [2/2]	1014
14.116	lpc::Out< T > Struct Template Reference	1014
14.116.1	Detailed Description	1014
14.117	lpc::Ret_array< T > Struct Template Reference	1015
14.117.1	Detailed Description	1015
14.118	lpc::Send_only Struct Reference	1016
14.118.1	Detailed Description	1016
14.119	lpc::Small_buf Class Reference	1016
14.119.1	Detailed Description	1017
14.119.2	Constructor & Destructor Documentation	1017
14.119.2.1	Small_buf() [1/2]	1017
14.119.2.2	Small_buf() [2/2]	1017
14.120	lpc::Snd_item Class Reference	1018
14.120.1	Detailed Description	1018
14.121	lpc::Str_cp_in< T > Class Template Reference	1018
14.121.1	Detailed Description	1019
14.121.2	Constructor & Destructor Documentation	1019
14.121.2.1	Str_cp_in()	1019
14.122	lpc::Varg Class Reference	1020
14.122.1	Detailed Description	1021
14.122.2	Member Function Documentation	1021
14.122.2.1	data()	1021
14.122.2.2	get_value()	1021
14.122.2.3	is_nil()	1022
14.122.2.4	is_of()	1022

14.122.2.5s_of_int()	1022
14.122.2.6length()	1023
14.122.2.7tag()	1023
14.122.2.8type()	1024
14.122.2.9value()	1024
14.1284::lpc::Varg_list< MAX > Class Template Reference	1025
14.123.1Detailed Description	1026
14.1244::lpc::Varg_list_ref Class Reference	1026
14.124.1Detailed Description	1027
14.124.2Constructor & Destructor Documentation	1027
14.124.2.1Varg_list_ref()	1027
14.1254::lpc_gate Class Reference	1028
14.125.1Detailed Description	1031
14.125.2Member Function Documentation	1031
14.125.2.1get_infos()	1031
14.1264::lpc_svr::Br_manager_no_buffers Class Reference	1031
14.126.1Detailed Description	1034
14.126.2Member Function Documentation	1034
14.126.2.1alloc_buffer_demand()	1034
14.1274::lpc_svr::Compound_reply Struct Reference	1035
14.127.1Detailed Description	1035
14.1284::lpc_svr::Default_loop_hooks Struct Reference	1036
14.128.1Detailed Description	1037
14.1294::lpc_svr::Default_setup_wait Struct Reference	1037
14.129.1Detailed Description	1038
14.1304::lpc_svr::Default_timeout Struct Reference	1038
14.130.1Detailed Description	1039
14.1314::lpc_svr::Direct_dispatch< R > Struct Template Reference	1039
14.131.1Detailed Description	1040
14.1324::lpc_svr::Direct_dispatch< R * > Struct Template Reference	1041

14.132.1 Detailed Description	1041
14.132.4 <code>lpc_svr::Exc_dispatch< R, Exc ></code> Struct Template Reference	1042
14.133.1 Detailed Description	1043
14.133.4 <code>lpc_svr::Ignore_errors</code> Struct Reference	1043
14.134.1 Detailed Description	1044
14.134.4 <code>lpc_svr::Server_iface</code> Class Reference	1044
14.135.1 Detailed Description	1047
14.135.2 Member Function Documentation	1047
14.135.2.1 <code>add_timeout()</code>	1047
14.135.2.2 <code>alloc_buffer_demand()</code>	1048
14.135.2.3 <code>get_rcv_cap()</code>	1048
14.135.2.4 <code>rcv_cap()</code> [1/2]	1049
14.135.2.5 <code>rcv_cap()</code> [2/2]	1050
14.135.2.6 <code>realloc_rcv_cap()</code>	1051
14.135.2.7 <code>remove_timeout()</code>	1052
14.136.4 <code>lpc_svr::Timed_work< HOOKS ></code> Class Template Reference	1053
14.136.1 Detailed Description	1053
14.137.4 <code>lpc_svr::Timeout</code> Class Reference	1053
14.137.1 Detailed Description	1055
14.137.2 Member Function Documentation	1055
14.137.2.1 <code>expired()</code>	1055
14.137.2.2 <code>timeout()</code>	1056
14.138.4 <code>lpc_svr::Timeout_queue</code> Class Reference	1056
14.138.1 Detailed Description	1057
14.138.2 Member Function Documentation	1057
14.138.2.1 <code>add()</code>	1057
14.138.2.2 <code>handle_expired_timeouts()</code>	1058
14.138.2.3 <code>next_timeout()</code>	1058
14.138.2.4 <code>remove()</code>	1058
14.138.2.5 <code>timeout_expired()</code>	1059

14.139.4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference	1060
14.139.4.1 Detailed Description	1062
14.139.4.2 Member Function Documentation	1062
14.139.4.2.1 add_timeout()	1062
14.139.4.2.2 remove_timeout()	1063
14.140.4::Irq Class Reference	1063
14.140.4.1 Detailed Description	1066
14.140.4.2 Member Function Documentation	1066
14.140.4.2.1 attach()	1066
14.140.4.2.2 detach()	1067
14.140.4.2.3 receive()	1068
14.140.4.2.4 unmask()	1069
14.140.4.2.5 wait()	1070
14.141.4::Irq_eoi Class Reference	1070
14.141.4.1 Detailed Description	1071
14.141.4.2 Member Function Documentation	1072
14.141.4.2.1 unmask()	1072
14.142.4::Irq_handler_object Struct Reference	1073
14.142.4.1 Detailed Description	1075
14.143.4::Irq_mux Struct Reference	1076
14.143.4.1 Detailed Description	1078
14.143.4.2 Member Function Documentation	1078
14.143.4.2.1 chain()	1078
14.144.4::Kip::Mem_desc Class Reference	1079
14.144.4.1 Detailed Description	1080
14.144.4.2 Member Enumeration Documentation	1081
14.144.4.2.1 Info_sub_type	1081
14.144.4.2.2 Mem_type	1081
14.144.4.3 Constructor & Destructor Documentation	1081
14.144.4.3.1 Mem_desc()	1081

14.144.4	Member Function Documentation	1082
14.144.4.1	all() [1/2]	1082
14.144.4.2	all() [2/2]	1083
14.144.4.3	count() [1/2]	1083
14.144.4.4	count() [2/2]	1084
14.144.4.5	end()	1085
14.144.4.6	first()	1085
14.144.4.7	is_virtual()	1086
14.144.4.8	set()	1086
14.144.4.9	size()	1087
14.144.4.10	start()	1087
14.144.4.11	sub_type()	1088
14.144.4.12	type()	1088
14.145.1	Kobject Class Reference	1088
14.145.1	Detailed Description	1089
14.145.2	Member Function Documentation	1089
14.145.2.1	cap()	1089
14.145.2.2	dec_refcnt()	1091
14.146.1	Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference	1092
14.146.1	Detailed Description	1093
14.146.2	Member Typedef Documentation	1094
14.146.2.1	__iface	1094
14.146.2.2	__iface_list	1094
14.146.2.3	Class	1095
14.146.3	Member Function Documentation	1095
14.146.3.1	__check_protocols__()	1095
14.146.3.2		1095
14.147.1	Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference	1096
14.147.1	Detailed Description	1097
14.147.2	Member Typedef Documentation	1098

14.147.2.1 <code>__lface</code>	1098
14.147.2.2 <code>__lface_list</code>	1098
14.147.2.3 <code>Class</code>	1099
14.147.3 Member Function Documentation	1099
14.147.3.1 <code>__check_protocols__()</code>	1099
14.147.3.2 <code>2d()</code>	1099
14.148.1 <code>Kobject_demand< T ></code> Struct Template Reference	1099
14.148.2 Detailed Description	1100
14.149.1 <code>Kobject_t< Derived, Base, PROTO, S_DEMAND ></code> Class Template Reference	1100
14.149.2 Detailed Description	1101
14.150.1 <code>Kobject_typeid< T ></code> Struct Template Reference	1102
14.150.2 Detailed Description	1102
14.150.3 Member Typedef Documentation	1103
14.150.3.1 <code>Demand</code>	1103
14.150.3.2 Member Function Documentation	1103
14.150.3.2.1 <code>demand()</code>	1103
14.150.3.2.2 <code>2d()</code>	1104
14.150.3.2.3 <code>proto_dispatch()</code>	1104
14.151.1 <code>Kobject_typeid< void ></code> Struct Template Reference	1105
14.151.2 Detailed Description	1106
14.152.1 <code>Kobject_x< Derived, ARGS ></code> Struct Template Reference	1106
14.152.2 Detailed Description	1107
14.153.1 <code>Meta Class Reference</code>	1107
14.153.2 Detailed Description	1109
14.153.3 Member Function Documentation	1110
14.153.3.1 <code>interface()</code>	1110
14.153.3.2 <code>num_interfaces()</code>	1110
14.153.3.3 <code>supports()</code>	1110
14.154.1 <code>Out_of_memory Class Reference</code>	1111
14.154.2 Detailed Description	1114

14.154::Pager Class Reference	1114
14.155.1Detailed Description	1116
14.155.2Member Function Documentation	1117
14.155.2.1page_fault()	1117
14.156::Platform_control Class Reference	1118
14.156.1Detailed Description	1120
14.156.2Member Enumeration Documentation	1120
14.156.2.1Opcode	1120
14.156.3Member Function Documentation	1120
14.156.3.1cpu_disable()	1120
14.156.3.2cpu_enable()	1121
14.156.3.3system_shutdown()	1121
14.156.3.4system_suspend()	1121
14.157::Poll_timeout_kipclock Class Reference	1122
14.157.1Detailed Description	1123
14.157.2Constructor & Destructor Documentation	1123
14.157.2.1Poll_timeout_kipclock()	1123
14.157.3Member Function Documentation	1123
14.157.3.1set()	1123
14.157.3.2test()	1124
14.157.3.3timed_out()	1125
14.158::Proto_t< P > Struct Template Reference	1125
14.158.1Detailed Description	1126
14.159::Rcv_endpoint Class Reference	1127
14.159.1Detailed Description	1129
14.159.2Member Function Documentation	1130
14.159.2.1bind_thread()	1130
14.160::Registry_iface Class Reference	1131
14.160.1Detailed Description	1132
14.160.2Member Function Documentation	1132

14.160.2.1	register_irq_obj()	1132
14.160.2.2	register_obj() [1/3]	1133
14.160.2.3	register_obj() [2/3]	1134
14.160.2.4	register_obj() [3/3]	1135
14.160.2.5	unregister_obj()	1135
14.161.4	Runtime_error Class Reference	1136
14.161.1	Detailed Description	1137
14.161.2	Constructor & Destructor Documentation	1138
14.161.2.1	Runtime_error()	1138
14.161.3	Member Function Documentation	1138
14.161.3.1	terr_no()	1138
14.161.3.2	extra_str()	1138
14.162.4	Scheduler Class Reference	1139
14.162.1	Detailed Description	1141
14.162.2	Member Function Documentation	1141
14.162.2.1	idle_time()	1141
14.162.2.2	info()	1142
14.162.2.3	is_online()	1143
14.162.2.4	run_thread()	1143
14.163.4	Semaphore Struct Reference	1144
14.163.1	Detailed Description	1147
14.163.2	Member Function Documentation	1147
14.163.2.1	down()	1147
14.163.2.2	up()	1148
14.164.4	Server< LOOP_HOOKS > Class Template Reference	1149
14.164.1	Detailed Description	1150
14.164.2	Constructor & Destructor Documentation	1150
14.164.2.1	Server()	1150
14.164.3	Member Function Documentation	1151
14.164.3.1	internal_loop()	1151

14.164.3.2oop()	1152
14.164::Server_object Class Reference	1152
14.165.1Detailed Description	1153
14.165.2Member Function Documentation	1153
14.165.2.1dispatch()	1153
14.164::Server_object_t< IFACE, BASE > Struct Template Reference	1154
14.166.1Detailed Description	1156
14.166.2Member Function Documentation	1156
14.166.2.1dispatch_meta_request()	1156
14.166.2.2get_buffer_demand()	1157
14.166.2.3proto_dispatch()	1157
14.164::Server_object_x< Derived, IFACE, BASE > Struct Template Reference	1158
14.167.1Detailed Description	1159
14.164::Smart_cap< T, SMART > Class Template Reference	1160
14.168.1Detailed Description	1163
14.168.2Constructor & Destructor Documentation	1163
14.168.2.1Smart_cap()	1163
14.164::String Class Reference	1164
14.169.1Detailed Description	1164
14.170::Task Class Reference	1164
14.170.1Detailed Description	1167
14.170.2Member Function Documentation	1167
14.170.2.1add_ku_mem()	1167
14.170.2.2cap_equal()	1168
14.170.2.3cap_has_child()	1168
14.170.2.4cap_valid()	1169
14.170.2.5delete_obj()	1169
14.170.2.6map()	1170
14.170.2.7release_cap()	1170
14.170.2.8unmap()	1171

14.170.2.9unmap_batch()	1172
14.171.4::Thread Class Reference	1173
14.171.1Detailed Description	1175
14.171.2Member Function Documentation	1176
14.171.2.1control()	1176
14.171.2.2ex_regs() [1/2]	1176
14.171.2.3ex_regs() [2/2]	1177
14.171.2.4modify_senders()	1178
14.171.2.5register_del_irq()	1179
14.171.2.6stats_time()	1179
14.171.2.7switch_to()	1180
14.171.2.8cpu_control()	1180
14.171.2.9cpu_control_ext()	1181
14.171.2.10cpu_resume_commit()	1182
14.171.2.11cpu_resume_start()	1182
14.172.4::Thread::Attr Class Reference	1183
14.172.1Detailed Description	1184
14.172.2Constructor & Destructor Documentation	1184
14.172.2.1Attr()	1184
14.172.3Member Function Documentation	1184
14.172.3.1bind()	1184
14.172.3.2exc_handler() [1/2]	1185
14.172.3.3exc_handler() [2/2]	1185
14.172.3.4pager() [1/2]	1185
14.172.3.5pager() [2/2]	1186
14.172.3.6ix_host_syscall()	1186
14.173.4::Thread::Modify_senders Class Reference	1186
14.173.1Detailed Description	1187
14.173.2Member Function Documentation	1187
14.173.2.1add()	1187

14.174::Triggerable Struct Reference	1188
14.174.1 Detailed Description	1190
14.174.2 Member Function Documentation	1190
14.174.2.1 trigger()	1191
14.175::Type_info Struct Reference	1192
14.175.1 Detailed Description	1192
14.176::Type_info::Demand Class Reference	1193
14.176.1 Detailed Description	1194
14.176.2 Constructor & Destructor Documentation	1194
14.176.2.1 Demand()	1195
14.176.3 Member Function Documentation	1195
14.176.3.1 no_demand()	1195
14.177::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference	1196
14.177.1 Detailed Description	1197
14.177.2 Member Enumeration Documentation	1198
14.177.2.1 anonymous enum	1198
14.178::Type_info::Demand_union_t< D1, D2 > Struct Template Reference	1198
14.178.1 Detailed Description	1200
14.179::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference	1201
14.179.1 Detailed Description	1202
14.180::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference	1202
14.180.1 Detailed Description	1203
14.181::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference	1203
14.181.1 Detailed Description	1204
14.182::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference	1204
14.182.1 Detailed Description	1205
14.183::Typeid::Detail::Rpc_end Struct Reference	1205
14.183.1 Detailed Description	1206
14.184::Typeid::P_dispatch< LIST > Struct Template Reference	1206
14.184.1 Detailed Description	1206

14.1854::Typeid::Raw_ipc< CLASS > Struct Template Reference	1207
14.185.1Detailed Description	1207
14.1864::Typeid::Rpc_nocode< OPERATION > Struct Template Reference	1207
14.186.1Detailed Description	1209
14.1874::Typeid::Rpc< RPCS > Struct Template Reference	1210
14.187.1Detailed Description	1211
14.1884::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference	1212
14.188.1Detailed Description	1212
14.1894::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference	1213
14.189.1Detailed Description	1214
14.1904::Typeid::Rpc_sys< ARG > Struct Template Reference	1215
14.190.1Detailed Description	1216
14.1914::Types::Bool< V > Struct Template Reference	1217
14.191.1Detailed Description	1218
14.1924::Types::False Struct Reference	1218
14.192.1Detailed Description	1220
14.1934::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference	1220
14.193.1Detailed Description	1222
14.193.2Member Enumeration Documentation	1223
14.193.2.1None_type	1223
14.193.3Constructor & Destructor Documentation	1223
14.193.3.1Flags() [1/2]	1223
14.193.3.2Flags() [2/2]	1224
14.193.4Member Function Documentation	1224
14.193.4.1clear()	1224
14.193.4.2from_raw()	1225
14.1944::Types::Same< A, B > Struct Template Reference	1225
14.194.1Detailed Description	1226
14.1954::Types::True Struct Reference	1227
14.195.1Detailed Description	1229

14.1964::Unknown_error Class Reference	1229
14.196.1 Detailed Description	1231
14.1974::Vcon Class Reference	1232
14.197.1 Detailed Description	1234
14.197.2 Member Function Documentation	1234
14.197.2.1 get_attr()	1234
14.197.2.2 read()	1235
14.197.2.3 read_with_flags()	1236
14.197.2.4 send()	1237
14.197.2.5 set_attr()	1237
14.197.2.6 write()	1238
14.1984::Vm Class Reference	1239
14.198.1 Detailed Description	1241
14.1994 _buf_regs_t Struct Reference	1242
14.199.1 Detailed Description	1242
14.2004 _exc_regs_t Struct Reference	1243
14.200.1 Detailed Description	1245
14.200.2 Field Documentation	1245
14.200.2.1 flags	1245
14.200.2.2 ss	1245
14.2014 _fpage_t Union Reference	1246
14.201.1 Detailed Description	1246
14.2024 _icu_info_t Struct Reference	1246
14.202.1 Detailed Description	1248
14.202.2 Field Documentation	1248
14.202.2.1 features	1248
14.2034 _icu_msi_info_t Struct Reference	1248
14.203.1 Detailed Description	1249
14.203.2 Field Documentation	1249
14.203.2.1 msi_data	1249

14.204. # <code>_kernel_info_mem_desc_t</code> Struct Reference	1249
14.204.1 Detailed Description	1250
14.205. # <code>_kernel_info_t</code> Struct Reference	1250
14.205.1 Detailed Description	1252
14.206. # <code>_msg_regs_t</code> Union Reference	1252
14.206.1 Detailed Description	1253
14.207. # <code>_msgtag_t</code> Struct Reference	1253
14.207.1 Detailed Description	1254
14.207.2 Member Function Documentation	1255
14.207.2.1 <code>flags()</code>	1255
14.208. # <code>_sched_cpu_set_t</code> Struct Reference	1255
14.208.1 Detailed Description	1256
14.208.2 Member Function Documentation	1256
14.208.2.1 <code>granularity()</code>	1256
14.208.2.2 <code>offset()</code>	1257
14.208.3 Field Documentation	1257
14.208.3.1 <code>gran_offset</code>	1257
14.209. # <code>_sched_param_t</code> Struct Reference	1258
14.209.1 Detailed Description	1258
14.210. # <code>_snd_fpage_t</code> Struct Reference	1259
14.210.1 Detailed Description	1259
14.211. # <code>_thread_regs_t</code> Struct Reference	1260
14.211.1 Detailed Description	1260
14.212. # <code>_timeout_s</code> Struct Reference	1261
14.212.1 Detailed Description	1261
14.213. # <code>_timeout_t</code> Union Reference	1262
14.213.1 Detailed Description	1262
14.214. # <code>_tracebuffer_status_t</code> Struct Reference	1263
14.214.1 Detailed Description	1264
14.214.2 Field Documentation	1264

14.214.2.1cnt_jobmap_tlb_flush	1264
14.215.1_tracebuffer_status_window_t Struct Reference	1265
14.215.1Detailed Description	1265
14.216.1_vcon_attr_t Struct Reference	1266
14.216.1Detailed Description	1266
14.217.1_vcpu_ipc_regs_t Struct Reference	1267
14.217.1Detailed Description	1267
14.218.1_vcpu_regs_t Struct Reference	1268
14.218.1Detailed Description	1269
14.218.2Field Documentation	1269
14.218.2.1ax	1270
14.218.2.2bp	1270
14.218.2.3bx	1270
14.218.2.4cx	1270
14.218.2.5di	1271
14.218.2.6dx	1271
14.218.2.7si	1271
14.219.1_vcpu_state_t Struct Reference	1272
14.219.1Detailed Description	1274
14.220.1_vhw_descriptor Struct Reference	1274
14.220.1Detailed Description	1275
14.220.2Field Documentation	1276
14.220.2.1count	1276
14.220.2.2descs	1276
14.220.2.3magic	1276
14.220.2.4version	1277
14.221.1_vhw_entry Struct Reference	1277
14.221.1Detailed Description	1278
14.221.2Field Documentation	1278
14.221.2.1fd	1278

14.221.2.2rq_no	1278
14.221.2.3mem_size	1278
14.221.2.4mem_start	1279
14.221.2.5provider_pid	1279
14.221.2.6type	1279
14.222_vm_svm_vmcb_control_area Struct Reference	1280
14.222.1Detailed Description	1280
14.223_vm_svm_vmcb_state_save_area Struct Reference	1280
14.223.1Detailed Description	1281
14.224_vm_svm_vmcb_state_save_area_seg Struct Reference	1282
14.224.1Detailed Description	1282
14.225_vm_svm_vmcb_t Struct Reference	1282
14.225.1Detailed Description	1283
14.226_vm_tz_state Struct Reference	1284
14.226.1Detailed Description	1284
14.227Re::Cap_alloc Class Reference	1284
14.227.1Detailed Description	1285
14.227.2Member Function Documentation	1285
14.227.2.1alloc() [1/2]	1286
14.227.2.2alloc() [2/2]	1286
14.227.2.3free()	1287
14.227.2.4get_cap_alloc()	1288
14.228Re::Console Class Reference	1289
14.228.1Detailed Description	1290
14.229Re::Dataspace Class Reference	1291
14.229.1Detailed Description	1293
14.229.2Member Enumeration Documentation	1293
14.229.2.1Map_flags	1293
14.229.3Member Function Documentation	1294
14.229.3.1allocate()	1294

14.229.3.2	<code>clear()</code>	1294
14.229.3.3	<code>copy_in()</code>	1295
14.229.3.4	<code>flags()</code>	1295
14.229.3.5	<code>info()</code>	1297
14.229.3.6	<code>map()</code>	1297
14.229.3.7	<code>map_region()</code>	1298
14.229.3.8	<code>phys()</code>	1299
14.229.3.9	<code>size()</code>	1300
14.230.4	<code>Re::Dataspace::Stats Struct Reference</code>	1300
14.230.1	Detailed Description	1301
14.231.4	<code>Re::Debug_obj Class Reference</code>	1301
14.231.1	Detailed Description	1302
14.231.2	Member Function Documentation	1303
14.231.2.1	<code>debug()</code>	1303
14.232.4	<code>Re::Dma_space Class Reference</code>	1303
14.232.1	Detailed Description	1304
14.232.2	Member Typedef Documentation	1304
14.232.2.1	Attributes	1305
14.232.3	Member Enumeration Documentation	1305
14.232.3.1	Attribute	1305
14.232.3.2	Direction	1305
14.232.3.3	Space_attrib	1306
14.232.4	Member Function Documentation	1306
14.232.4.1	<code>associate()</code>	1306
14.232.4.2	<code>disassociate()</code>	1307
14.232.4.3	<code>map()</code>	1307
14.232.4.4	<code>unmap()</code>	1308
14.233.4	<code>Re::Env Class Reference</code>	1308
14.233.1	Detailed Description	1310
14.233.2	Member Function Documentation	1311

14.233.2.1env()	1311
14.233.2.2factory() [1/2]	1312
14.233.2.3factory() [2/2]	1312
14.233.2.4first_free_cap() [1/2]	1312
14.233.2.5first_free_cap() [2/2]	1312
14.233.2.6first_free_utcb() [1/2]	1313
14.233.2.7first_free_utcb() [2/2]	1313
14.233.2.8get()	1313
14.233.2.9get_cap() [1/2]	1314
14.233.2.10get_cap() [2/2]	1315
14.233.2.11initial_caps() [1/2]	1315
14.233.2.12initial_caps() [2/2]	1315
14.233.2.13log() [1/2]	1316
14.233.2.14log() [2/2]	1316
14.233.2.15main_thread() [1/2]	1316
14.233.2.16main_thread() [2/2]	1317
14.233.2.17mem_alloc() [1/2]	1317
14.233.2.18mem_alloc() [2/2]	1317
14.233.2.19parent() [1/2]	1318
14.233.2.20parent() [2/2]	1318
14.233.2.21m() [1/2]	1318
14.233.2.22m() [2/2]	1319
14.233.2.23scheduler() [1/2]	1319
14.233.2.24scheduler() [2/2]	1319
14.233.2.25ask()	1320
14.233.2.26utcb_area() [1/2]	1320
14.233.2.27utcb_area() [2/2]	1320
14.234.1Re::Event Class Reference	1321
14.234.1.Detailed Description	1324
14.234.2Member Function Documentation	1324

14.234.2.1get_buffer()	1324
14.2354Re::Event_buffer_t< PAYLOAD > Class Template Reference	1325
14.235.1Detailed Description	1326
14.235.2Constructor & Destructor Documentation	1326
14.235.2.1Event_buffer_t()	1326
14.235.3Member Function Documentation	1327
14.235.3.1next()	1327
14.235.3.2put()	1327
14.2364Re::Event_buffer_t< PAYLOAD >::Event Struct Reference	1328
14.236.1Detailed Description	1329
14.2374Re::Inhibitor Class Reference	1329
14.237.1Detailed Description	1332
14.237.2Member Enumeration Documentation	1332
14.237.2.1anonymous enum	1332
14.237.3Member Function Documentation	1332
14.237.3.1acquire()	1332
14.237.3.2next_lock_info()	1333
14.237.3.3release()	1334
14.2384Re::Log Class Reference	1334
14.238.1Detailed Description	1337
14.238.2Member Function Documentation	1337
14.238.2.1print()	1337
14.238.2.2println()	1337
14.2394Re::Mem_alloc Class Reference	1337
14.239.1Detailed Description	1340
14.239.2Member Enumeration Documentation	1340
14.239.2.1Mem_alloc_flags	1340
14.239.3Member Function Documentation	1341
14.239.3.1alloc()	1341
14.239.3.2free()	1342

14.240.4	Re::Mmio_space Struct Reference	1342
14.240.1	Detailed Description	1345
14.240.2	Member Enumeration Documentation	1345
14.240.2.1	Access_width	1345
14.240.3	Member Function Documentation	1345
14.240.3.1	mmio_read()	1346
14.240.3.2	mmio_write()	1346
14.241.4	Re::Namespace Class Reference	1347
14.241.1	Detailed Description	1349
14.241.2	Member Enumeration Documentation	1349
14.241.2.1	Query_result_flags	1349
14.241.2.2	Register_flags	1349
14.241.3	Member Function Documentation	1350
14.241.3.1	query() [1/2]	1350
14.241.3.2	query() [2/2]	1351
14.241.3.3	register_obj()	1352
14.241.3.4	unlink()	1352
14.242.4	Re::Ned::Cmd_control Class Reference	1353
14.242.1	Detailed Description	1354
14.242.2	Member Function Documentation	1354
14.242.2.1	execute() [1/2]	1354
14.242.2.2	execute() [2/2]	1354
14.243.4	Re::Parent Class Reference	1355
14.243.1	Detailed Description	1356
14.243.2	Member Function Documentation	1357
14.243.2.1	signal()	1357
14.244.4	Re::Rm Class Reference	1357
14.244.1	Detailed Description	1360
14.244.2	Member Enumeration Documentation	1361
14.244.2.1	Attach_flags	1361

14.244.2.2	Detach_flags	1361
14.244.2.3	Detach_result	1361
14.244.2.4	Region_flags	1362
14.244.3	Member Function Documentation	1362
14.244.3.1	attach() [1/2]	1362
14.244.3.2	attach() [2/2]	1364
14.244.3.3	detach() [1/3]	1365
14.244.3.4	detach() [2/3]	1366
14.244.3.5	detach() [3/3]	1366
14.244.3.6	find()	1367
14.244.3.7	free_area()	1368
14.244.3.8	reserve_area() [1/2]	1368
14.244.3.9	reserve_area() [2/2]	1369
14.245	Re::Smart_cap_auto< Unmap_flags > Class Template Reference	1370
14.245.1	Detailed Description	1371
14.246	Re::Smart_count_cap< Unmap_flags > Class Template Reference	1371
14.246.1	Detailed Description	1372
14.247	Re::Util::Auto_cap< T > Struct Template Reference	1372
14.247.1	Detailed Description	1372
14.248	Re::Util::Auto_del_cap< T > Struct Template Reference	1373
14.248.1	Detailed Description	1373
14.249	Re::Util::Br_manager Class Reference	1374
14.249.1	Detailed Description	1377
14.249.2	Member Function Documentation	1377
14.249.2.1	alloc_buffer_demand()	1377
14.249.2.2	get_rcv_cap()	1377
14.249.2.3	realloc_rcv_cap()	1378
14.249.2.4	set_rcv_cap_flags()	1379
14.250	Re::Util::Br_manager_hooks Struct Reference	1379
14.250.1	Detailed Description	1380

14.251.4Re::Util::Br_manager_timeout_hooks Struct Reference	1380
14.251.1Detailed Description	1382
14.252.4Re::Util::Cap_alloc_base Class Reference	1383
14.252.1Detailed Description	1383
14.253.4Re::Util::Counter< COUNTER > Struct Template Reference	1384
14.253.1Detailed Description	1384
14.254.4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference	1384
14.254.1Detailed Description	1386
14.254.2Constructor & Destructor Documentation	1386
14.254.2.1Counting_cap_alloc()	1386
14.254.3Member Function Documentation	1386
14.254.3.1alloc() [1/2]	1387
14.254.3.2alloc() [2/2]	1388
14.254.3.3free()	1388
14.254.3.4release()	1389
14.254.3.5setup()	1390
14.254.3.6take()	1391
14.255.4Re::Util::Dataspace_svr Class Reference	1391
14.255.1Detailed Description	1393
14.255.2Member Function Documentation	1393
14.255.2.1allocate()	1393
14.255.2.2clear()	1394
14.255.2.3copy()	1395
14.255.2.4is_static()	1396
14.255.2.5map()	1397
14.255.2.6map_hook()	1397
14.255.2.7page_shift()	1398
14.255.2.8phys()	1399
14.255.2.9release()	1400
14.255.2.10take()	1401

14.2564Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference	1401
14.256.1Detailed Description	1403
14.256.2Member Function Documentation	1404
14.256.2.1foreach_available_event()	1404
14.256.2.2process()	1404
14.2574Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference	1405
14.257.1Detailed Description	1407
14.257.2Member Function Documentation	1407
14.257.2.1attach()	1408
14.257.2.2buf()	1408
14.257.2.3detach()	1408
14.2584Re::Util::Event_svr< SVR > Class Template Reference	1409
14.258.1Detailed Description	1411
14.2594Re::Util::Event_t< PAYLOAD > Class Template Reference	1411
14.259.1Detailed Description	1412
14.259.2Member Enumeration Documentation	1412
14.259.2.1Mode	1412
14.259.3Member Function Documentation	1412
14.259.3.1buffer()	1413
14.259.3.2nit()	1413
14.259.3.3nit_poll()	1414
14.259.3.4rq()	1415
14.2604Re::Util::Item_alloc_base Class Reference	1415
14.260.1Detailed Description	1416
14.2614Re::Util::Names::Name Class Reference	1416
14.261.1Detailed Description	1417
14.2624Re::Util::Names::Name_space Class Reference	1417
14.262.1Detailed Description	1418
14.262.2Member Function Documentation	1419
14.262.2.1alloc_dynamic_entry()	1419

14.262.2.2	copy_receive_cap()	1419
14.262.2.3	free_capability()	1419
14.262.2.4	free_dynamic_entry()	1420
14.262.2.5	free_epiface()	1420
14.262.2.6	get_epiface()	1420
14.263.4	Re::Util::Object_registry Class Reference	1421
14.263.1	Detailed Description	1423
14.263.2	Constructor & Destructor Documentation	1423
14.263.2.1	Object_registry() [1/2]	1423
14.263.2.2	Object_registry() [2/2]	1423
14.263.3	Member Function Documentation	1424
14.263.3.1	register_irq_obj()	1424
14.263.3.2	register_obj() [1/3]	1425
14.263.3.3	register_obj() [2/3]	1425
14.263.3.4	register_obj() [3/3]	1426
14.263.3.5	unregister_obj()	1426
14.264.4	Re::Util::Ref_cap< T > Struct Template Reference	1427
14.264.1	Detailed Description	1427
14.265.4	Re::Util::Ref_del_cap< T > Struct Template Reference	1428
14.265.1	Detailed Description	1428
14.266.4	Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference	1429
14.266.1	Detailed Description	1431
14.266.2	Constructor & Destructor Documentation	1432
14.266.2.1	Registry_server() [1/2]	1432
14.266.2.2	Registry_server() [2/2]	1432
14.266.3	Member Function Documentation	1432
14.266.3.1	registry() [1/2]	1433
14.266.3.2	registry() [2/2]	1433
14.267.4	Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference	1433
14.267.1	Detailed Description	1434

14.268	Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference	1434
14.268.1	Detailed Description	1434
14.269	Re::Util::Vcon_svr< SVR > Class Template Reference	1435
14.269.1	Detailed Description	1435
14.270	Re::Util::Video::Goos_svr Class Reference	1436
14.270.1	Detailed Description	1437
14.270.2	Member Function Documentation	1437
14.270.2.1	get_fb()	1437
14.270.2.2	nit_infos()	1438
14.270.2.3	refresh()	1438
14.270.2.4	screen_info()	1439
14.270.2.5	view_info()	1439
14.271	Re::Vfs::Be_file Class Reference	1440
14.271.1	Detailed Description	1442
14.271.2	Member Function Documentation	1443
14.271.2.1	data_space()	1443
14.271.2.2	stat64()	1443
14.271.2.3	unlock_all_locks()	1444
14.272	Re::Vfs::Be_file_system Class Reference	1444
14.272.1	Detailed Description	1445
14.272.2	Constructor & Destructor Documentation	1446
14.272.2.1	Be_file_system()	1446
14.272.2.2	~Be_file_system()	1446
14.272.3	Member Function Documentation	1446
14.272.3.1	type()	1446
14.273	Re::Vfs::Directory Class Reference	1447
14.273.1	Detailed Description	1448
14.273.2	Member Function Documentation	1449
14.273.2.1	faccessat()	1449
14.273.2.2	link()	1449

14.273.2.3mkdir()	1450
14.273.2.4rename()	1450
14.273.2.5mkdir()	1451
14.273.2.6symlink()	1451
14.273.2.7unlink()	1451
14.274Re::Vfs::File Class Reference	1452
14.274.1Detailed Description	1453
14.275Re::Vfs::File_system Class Reference	1454
14.275.1Detailed Description	1455
14.275.2Member Function Documentation	1455
14.275.2.1mount()	1455
14.275.2.2type()	1456
14.276Re::Vfs::Fs Class Reference	1456
14.276.1Detailed Description	1458
14.276.2Member Function Documentation	1459
14.276.2.1alloc_fd()	1459
14.276.2.2free_fd()	1459
14.276.2.3get_file()	1460
14.276.2.4mount()	1460
14.276.2.5set_fd()	1460
14.277Re::Vfs::Generic_file Class Reference	1461
14.277.1Detailed Description	1463
14.277.2Member Function Documentation	1463
14.277.2.1fchmod()	1464
14.277.2.2stat64()	1464
14.277.2.3get_status_flags()	1464
14.277.2.4set_status_flags()	1464
14.277.2.5unlock_all_locks()	1465
14.278Re::Vfs::Mman Class Reference	1466
14.278.1Detailed Description	1467

14.274	L4Re::Vfs::Ops Class Reference	1467
14.279	1.Detailed Description	1469
14.280	L4Re::Vfs::Regular_file Class Reference	1470
14.280	1.Detailed Description	1471
14.280	2.Member Function Documentation	1472
14.280.2	1.data_space()	1472
14.280.2	2.datasync()	1472
14.280.2	3.sync()	1472
14.280.2	4.truncate64()	1472
14.280.2	5.get_lock()	1473
14.280.2	6.seek64()	1473
14.280.2	7.readv()	1474
14.280.2	8.set_lock()	1474
14.280.2	9.writev()	1474
14.281	L4Re::Vfs::Special_file Class Reference	1475
14.281	1.Detailed Description	1477
14.281	2.Member Function Documentation	1477
14.281.2	1.ioctl()	1477
14.282	L4Re::Video::Color_component Class Reference	1478
14.282	1.Detailed Description	1479
14.282	2.Constructor & Destructor Documentation	1479
14.282.2	1.Color_component()	1479
14.282	3.Member Function Documentation	1479
14.282.3	1.dump()	1479
14.282.3	2.get()	1480
14.282.3	3.operator==()	1480
14.282.3	4.set()	1480
14.282.3	5.shift()	1481
14.282.3	6.size()	1481
14.283	L4Re::Video::Goos Class Reference	1482

14.283.1Detailed Description	1484
14.283.2Member Enumeration Documentation	1484
14.283.2.1Flags	1484
14.283.3Member Function Documentation	1485
14.283.3.1create_buffer()	1485
14.283.3.2create_view()	1485
14.283.3.3delete_buffer()	1486
14.283.3.4delete_view()	1486
14.283.3.5get_static_buffer()	1486
14.283.3.6info()	1487
14.283.3.7view()	1487
14.284Re::Video::Goos::Info Struct Reference	1488
14.284.1Detailed Description	1489
14.284.2Member Function Documentation	1489
14.284.2.1auto_refresh()	1489
14.285Re::Video::Pixel_info Class Reference	1490
14.285.1Detailed Description	1491
14.285.2Constructor & Destructor Documentation	1491
14.285.2.1Pixel_info() [1/2]	1491
14.285.2.2Pixel_info() [2/2]	1492
14.285.3Member Function Documentation	1492
14.285.3.1a() [1/2]	1492
14.285.3.2a() [2/2]	1492
14.285.3.3b() [1/2]	1493
14.285.3.4b() [2/2]	1493
14.285.3.5bits_per_pixel()	1493
14.285.3.6bytes_per_pixel() [1/2]	1494
14.285.3.7bytes_per_pixel() [2/2]	1494
14.285.3.8dump()	1494
14.285.3.9g() [1/2]	1495

14.285.3.10() [2/2]	1495
14.285.3.11Has_alpha()	1496
14.285.3.12operator==()	1496
14.285.3.13() [1/2]	1497
14.285.3.14() [2/2]	1497
14.286.1Re::Video::View Class Reference	1497
14.286.1Detailed Description	1499
14.286.2Member Enumeration Documentation	1499
14.286.2.1Flags	1499
14.286.2.2V_flags	1499
14.286.3Member Function Documentation	1500
14.286.3.1info()	1500
14.286.3.2refresh()	1500
14.286.3.3set_info()	1501
14.286.3.4set_viewport()	1501
14.286.3.5stack()	1502
14.287.1Re::Video::View::Info Struct Reference	1502
14.287.1Detailed Description	1504
14.288.1re_aux_t Struct Reference	1505
14.288.1Detailed Description	1505
14.289.1re_ds_stats_t Struct Reference	1506
14.289.1Detailed Description	1506
14.290.1re_elf_aux_mword_t Struct Reference	1506
14.290.1Detailed Description	1507
14.291.1re_elf_aux_t Struct Reference	1507
14.291.1Detailed Description	1508
14.292.1re_elf_aux_vma_t Struct Reference	1508
14.292.1Detailed Description	1508
14.293.1re_env_cap_entry_t Struct Reference	1509
14.293.1Detailed Description	1509

14.293.2	Constructor & Destructor Documentation	1509
14.293.2.1	l4re_env_cap_entry_t()	1510
14.293.3	Field Documentation	1510
14.293.3.1	flags	1510
14.294	l4re_env_t Struct Reference	1511
14.294.1	Detailed Description	1512
14.295	l4re_event_t Struct Reference	1512
14.295.1	Detailed Description	1513
14.296	l4re_video_color_component_t Struct Reference	1513
14.296.1	Detailed Description	1514
14.297	l4re_video_goos_info_t Struct Reference	1514
14.297.1	Detailed Description	1515
14.298	l4re_video_pixel_info_t Struct Reference	1515
14.298.1	Detailed Description	1516
14.299	l4re_video_view_info_t Struct Reference	1516
14.299.1	Detailed Description	1518
14.300	l4re_video_view_t Struct Reference	1518
14.300.1	Detailed Description	1518
14.301	l4util_idt_desc_t Struct Reference	1519
14.301.1	Detailed Description	1519
14.302	l4util_idt_header_t Struct Reference	1520
14.302.1	Detailed Description	1520
14.303	l4util_mb_addr_range_t Struct Reference	1521
14.303.1	Detailed Description	1521
14.304	l4util_mb_apm_t Struct Reference	1522
14.304.1	Detailed Description	1522
14.305	l4util_mb_drive_t Struct Reference	1522
14.305.1	Detailed Description	1523
14.305.2	Field Documentation	1523
14.305.2.1	drive_cylinders	1523

14.305.2.2	drive_mode	1524
14.305.2.3	drive_number	1524
14.306	util_mb_info_t Struct Reference	1524
14.306.1	Detailed Description	1526
14.307	util_mb_mod_t Struct Reference	1526
14.307.1	Detailed Description	1526
14.307.2	Field Documentation	1527
14.307.2.1	mod_end	1527
14.307.2.2	mod_start	1527
14.308	util_mb_vbe_ctrl_t Struct Reference	1527
14.308.1	Detailed Description	1528
14.309	util_mb_vbe_mode_t Struct Reference	1528
14.309.1	Detailed Description	1530
14.310	lvbus::Device Class Reference	1531
14.310.1	Detailed Description	1533
14.310.2	Member Function Documentation	1533
14.310.2.1	bus_cap()	1533
14.310.2.2	dev_handle()	1534
14.310.2.3	device()	1535
14.310.2.4	device_by_hid()	1536
14.310.2.5	get_resource()	1537
14.310.2.6	is_compatible()	1538
14.310.2.7	next_device()	1538
14.310.2.8	operator!=(())	1539
14.310.2.9	operator==(())	1539
14.310.3	Field Documentation	1539
14.310.3.1	_bus	1540
14.311	lvbus::Gpio_module Class Reference	1540
14.311.1	Detailed Description	1543
14.311.2	Member Function Documentation	1543

14.311.2.1	<code>config_pad()</code>	1543
14.311.2.2	<code>get()</code>	1544
14.311.2.3	<code>pin()</code>	1544
14.311.2.4	<code>set()</code>	1545
14.311.2.5	<code>setup()</code>	1546
14.312	<code>vbus::Gpio_module::Pin_slice Struct Reference</code>	1546
14.312.1	Detailed Description	1547
14.313	<code>vbus::Gpio_pin Class Reference</code>	1547
14.313.1	Detailed Description	1550
14.313.2	Member Function Documentation	1550
14.313.2.1	<code>config_get()</code>	1550
14.313.2.2	<code>config_pad()</code>	1551
14.313.2.3	<code>config_pull()</code>	1551
14.313.2.4	<code>get()</code>	1552
14.313.2.5	<code>pin()</code>	1553
14.313.2.6	<code>set()</code>	1553
14.313.2.7	<code>setup()</code>	1553
14.313.2.8	<code>to_irq()</code>	1554
14.314	<code>vbus::Icu Class Reference</code>	1555
14.314.1	Detailed Description	1557
14.315	<code>vbus::Pci_dev Class Reference</code>	1557
14.315.1	Detailed Description	1560
14.315.2	Member Function Documentation	1560
14.315.2.1	<code>cfg_read()</code>	1560
14.315.2.2	<code>cfg_write()</code>	1561
14.315.2.3	<code>irq_enable()</code>	1561
14.316	<code>vbus::Pci_host_bridge Class Reference</code>	1562
14.316.1	Detailed Description	1565
14.316.2	Member Function Documentation	1565
14.316.2.1	<code>cfg_read()</code>	1565

14.316.2.2	xfg_write()	1566
14.316.2.3	irq_enable()	1567
14.317	Vbus::Pm< DEC > Class Template Reference	1568
14.317.1	Detailed Description	1569
14.318	Vbus::Vbus Class Reference	1569
14.318.1	Detailed Description	1572
14.318.2	Member Function Documentation	1572
14.318.2.1	assign_dma_domain() [1/2]	1572
14.318.2.2	assign_dma_domain() [2/2]	1573
14.318.2.3	release_resource()	1574
14.318.2.4	request_resource()	1575
14.318.2.5	root()	1575
14.319	vbus_device_t Struct Reference	1576
14.319.1	Detailed Description	1576
14.320	vbus_resource_t Struct Reference	1577
14.320.1	Detailed Description	1577
14.321	Vcpu::State Class Reference	1578
14.321.1	Detailed Description	1578
14.321.2	Constructor & Destructor Documentation	1578
14.321.2.1	State()	1578
14.321.3	Member Function Documentation	1579
14.321.3.1	add()	1579
14.321.3.2	clear()	1579
14.321.3.3	set()	1579
14.322	Vcpu::Vcpu Class Reference	1580
14.322.1	Detailed Description	1584
14.322.2	Member Function Documentation	1584
14.322.2.1	icast() [1/2]	1584
14.322.2.2	icast() [2/2]	1584
14.322.2.3	entry_ip()	1585

14.322.2.4	entry_sp()	1585
14.322.2.5	ext_alloc()	1586
14.322.2.6	() [1/2]	1586
14.322.2.7	() [2/2]	1586
14.322.2.8	rq_disable_save()	1587
14.322.2.9	rq_enable()	1587
14.322.2.10	rq_restore()	1588
14.322.2.11	irq_entry()	1588
14.322.2.12	page_fault_entry()	1589
14.322.2.13	() [1/2]	1590
14.322.2.14	() [2/2]	1590
14.322.2.15	saved_state() [1/2]	1590
14.322.2.16	saved_state() [2/2]	1591
14.322.2.17	state() [1/2]	1591
14.322.2.18	state() [2/2]	1591
14.322.2.19	task()	1591
14.322.2.20	wait_for_event()	1592
14.323.4	virtio::Device Class Reference	1593
14.323.1	Detailed Description	1595
14.323.2	Member Function Documentation	1595
14.323.2.1	config_queue()	1595
14.323.2.2	device_config()	1596
14.323.2.3	device_notification_irq()	1596
14.323.2.4	register_ds()	1597
14.323.2.5	register_iface()	1597
14.323.2.6	set_status()	1598
14.324.4	virtio::Driver::Virtqueue Class Reference	1598
14.324.1	Detailed Description	1601
14.324.2	Member Function Documentation	1601
14.324.2.1	alloc_descriptor()	1601

14.324.2.2desc()	1601
14.324.2.3enqueue_descriptor()	1602
14.324.2.4find_next_used()	1602
14.324.2.5free_descriptor()	1602
14.324.2.6init_queue() [1/2]	1603
14.324.2.7init_queue() [2/2]	1603
14.324.2.8initialize_rings()	1604
14.325virtio::Ptr< T > Class Template Reference	1604
14.325.1Detailed Description	1606
14.325.2Member Enumeration Documentation	1606
14.325.2.1Invalid_type	1606
14.325.3Member Function Documentation	1606
14.325.3.1get()	1606
14.325.3.2s_valid()	1607
14.326virtio::Svr::Bad_descriptor Struct Reference	1608
14.326.1Detailed Description	1609
14.326.2Member Enumeration Documentation	1609
14.326.2.1Error	1609
14.326.3Constructor & Destructor Documentation	1609
14.326.3.1Bad_descriptor()	1609
14.326.4Member Function Documentation	1610
14.326.4.1message()	1610
14.327virtio::Svr::Block_dev< Ds_data > Class Template Reference	1610
14.327.1Detailed Description	1613
14.327.2Constructor & Destructor Documentation	1614
14.327.2.1Block_dev()	1614
14.327.3Member Function Documentation	1614
14.327.3.1finalize_request()	1614
14.327.3.2get_writeback()	1615
14.327.3.3process_request()	1615

14.327.3.4	register_obj()	1615
14.327.3.5	reset_client()	1616
14.327.3.6	set_blk_size()	1617
14.327.3.7	set_config_wce()	1617
14.327.3.8	set_discard()	1617
14.327.3.9	set_flush()	1618
14.327.3.10	set_size_max()	1618
14.327.3.11	set_topology()	1618
14.327.3.12	set_write_zeroes()	1619
14.328.1	virtio::Svr::Block_request< Ds_data > Class Template Reference	1619
14.328.1	Detailed Description	1620
14.328.2	Member Function Documentation	1620
14.328.2.1	data_size()	1620
14.328.2.2	next_block()	1621
14.329.1	virtio::Svr::Data_buffer Struct Reference	1621
14.329.1	Detailed Description	1622
14.329.2	Constructor & Destructor Documentation	1623
14.329.2.1	Data_buffer()	1623
14.329.3	Member Function Documentation	1623
14.329.3.1	copy_to()	1623
14.329.3.2	done()	1624
14.329.3.3	set()	1624
14.329.3.4	skip()	1625
14.330.1	virtio::Svr::Dev_config Class Reference	1626
14.330.1	Detailed Description	1627
14.330.2	Constructor & Destructor Documentation	1627
14.330.2.1	Dev_config()	1627
14.330.3	Member Function Documentation	1628
14.330.3.1	change_queue_config()	1628
14.330.3.2	ds()	1629

14.330.3.3	get_cmd()	1630
14.330.3.4	guest_features()	1630
14.330.3.5	hdr()	1631
14.330.3.6	negotiated_features()	1631
14.330.3.7	qconfig()	1632
14.330.3.8	reset_cmd()	1633
14.330.3.9	reset_queue()	1633
14.330.3.10	set_failed()	1635
14.330.3.11	set_status()	1635
14.330.3.12	status()	1636
14.331.1	virtio::Svr::Dev_features Struct Reference	1637
14.331.1	Detailed Description	1638
14.331.2	Member Typedef Documentation	1638
14.331.2.1	ring_event_idx_bfm_t	1638
14.331.2.2	ring_indirect_desc_bfm_t	1638
14.331.3	Member Function Documentation	1639
14.331.3.1	ring_event_idx() [1/2]	1639
14.331.3.2	ring_event_idx() [2/2]	1639
14.331.3.3	ring_indirect_desc() [1/2]	1639
14.331.3.4	ring_indirect_desc() [2/2]	1639
14.332.1	virtio::Svr::Dev_status Struct Reference	1640
14.332.1	Detailed Description	1641
14.332.2	Member Typedef Documentation	1641
14.332.2.1	lacked_bfm_t	1641
14.332.2.2	driver_bfm_t	1642
14.332.2.3	driver_ok_bfm_t	1642
14.332.2.4	failed_bfm_t	1642
14.332.2.5	features_ok_bfm_t	1642
14.332.3	Member Function Documentation	1642
14.332.3.1	lacked() [1/2]	1643

14.332.3.2acked() [2/2]	1643
14.332.3.3driver() [1/2]	1643
14.332.3.4driver() [2/2]	1644
14.332.3.5driver_ok() [1/2]	1644
14.332.3.6driver_ok() [2/2]	1644
14.332.3.7failed() [1/2]	1645
14.332.3.8failed() [2/2]	1645
14.332.3.9features_ok() [1/2]	1645
14.332.3.10features_ok() [2/2]	1645
14.332.3.11running()	1646
14.333.4virtio::Svr::Device_t< DATA > Class Template Reference	1646
14.333.1Detailed Description	1649
14.333.2Member Function Documentation	1649
14.333.2.1device_error()	1649
14.333.2.2device_notify_irq()	1650
14.333.2.3handle_mem_cmd_write()	1650
14.333.2.4init_mem_info()	1650
14.333.2.5register_driver_irq()	1650
14.333.2.6reset_queue_config()	1651
14.333.2.7setup_queue()	1651
14.334.4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference	1652
14.334.1Detailed Description	1654
14.334.2Member Function Documentation	1654
14.334.2.1add()	1654
14.334.2.2find()	1655
14.334.2.3full()	1656
14.334.2.4init()	1656
14.334.2.5load_desc() [1/3]	1657
14.334.2.6load_desc() [2/3]	1657
14.334.2.7load_desc() [3/3]	1658

14.334.2.remove()	1659
14.335.4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference	1659
14.335.1Detailed Description	1661
14.335.2Constructor & Destructor Documentation	1662
14.335.2.1Driver_mem_region_t()	1662
14.335.3Member Function Documentation	1662
14.335.3.1contains()	1662
14.335.3.2drv_base()	1663
14.335.3.3ds()	1663
14.335.3.4ds_offset()	1663
14.335.3.5empty()	1664
14.335.3.6flags()	1664
14.335.3.7is_writable()	1664
14.335.3.8local()	1664
14.335.3.9local_base()	1665
14.335.3.10size()	1666
14.336.4virtio::Svr::Request_processor Class Reference	1666
14.336.1Detailed Description	1667
14.336.2Member Function Documentation	1667
14.336.2.1current_flags()	1667
14.336.2.2has_more()	1668
14.336.2.3next()	1668
14.336.2.4start() [1 / 2]	1670
14.336.2.5start() [2 / 2]	1671
14.337.4virtio::Svr::Virtqueue Class Reference	1672
14.337.1Detailed Description	1675
14.337.2Member Function Documentation	1675
14.337.2.1consumed()	1675
14.337.2.2desc()	1676
14.337.2.3desc_avail()	1677

14.337.2.4	disable_notify()	1677
14.337.2.5	enable_notify()	1677
14.337.2.6	next_avail()	1678
14.338.4	virtio::Svr::Virtqueue::Head_desc Class Reference	1679
14.338.1	Detailed Description	1679
14.338.2	Member Function Documentation	1679
14.338.2.1	desc()	1680
14.338.2.2	operator Null_ptr_check const *()	1680
14.338.2.3	valid()	1680
14.339.4	virtio::Virtqueue Class Reference	1681
14.339.1	Detailed Description	1684
14.339.2	Member Function Documentation	1684
14.339.2.1	avail_align()	1684
14.339.2.2	avail_size()	1684
14.339.2.3	desc_align()	1685
14.339.2.4	desc_size()	1685
14.339.2.5	disable()	1686
14.339.2.6	dump()	1687
14.339.2.7	get_avail_idx()	1687
14.339.2.8	get_tail_avail_idx()	1688
14.339.2.9	no_notify_guest()	1688
14.339.2.10	no_notify_host() [1/2]	1689
14.339.2.11	no_notify_host() [2/2]	1689
14.339.2.12	um()	1690
14.339.2.13	ready()	1690
14.339.2.14	setup()	1691
14.339.2.15	setup_simple()	1691
14.339.2.16	total_size() [1/2]	1692
14.339.2.17	total_size() [2/2]	1692
14.339.2.18	used_align()	1693

14.339.2.1	used_size()	1693
14.340.1	virtio::Virtqueue::Avail Class Reference	1694
14.340.1	Detailed Description	1695
14.341.1	virtio::Virtqueue::Avail::Flags Struct Reference	1695
14.341.1	Detailed Description	1696
14.341.2	Member Typedef Documentation	1696
14.341.2.1	no_irq_bfm_t	1696
14.341.3	Member Function Documentation	1696
14.341.3.1	no_irq() [1/2]	1696
14.341.3.2	no_irq() [2/2]	1697
14.342.1	virtio::Virtqueue::Desc Class Reference	1697
14.342.1	Detailed Description	1699
14.343.1	virtio::Virtqueue::Desc::Flags Struct Reference	1699
14.343.1	Detailed Description	1700
14.343.2	Member Typedef Documentation	1700
14.343.2.1	indirect_bfm_t	1701
14.343.2.2	next_bfm_t	1701
14.343.2.3	write_bfm_t	1701
14.343.3	Member Function Documentation	1701
14.343.3.1	indirect() [1/2]	1701
14.343.3.2	indirect() [2/2]	1702
14.343.3.3	next() [1/2]	1702
14.343.3.4	next() [2/2]	1702
14.343.3.5	write() [1/2]	1702
14.343.3.6	write() [2/2]	1703
14.344.1	virtio::Virtqueue::Used Class Reference	1703
14.344.1	Detailed Description	1704
14.345.1	virtio::Virtqueue::Used::Flags Struct Reference	1704
14.345.1	Detailed Description	1705
14.345.2	Member Typedef Documentation	1705

14.345.2.1no_notify_bfm_t	1705
14.345.3Member Function Documentation	1705
14.345.3.1no_notify() [1/2]	1706
14.345.3.2no_notify() [2/2]	1706
14.346virtio::Virtqueue::Used_elem Struct Reference	1706
14.346.1Detailed Description	1707
14.346.2Constructor & Destructor Documentation	1707
14.346.2.1Used_elem()	1707
14.347virtio_block_config_t Struct Reference	1707
14.347.1Detailed Description	1708
14.347.2Field Documentation	1708
14.347.2.1blk_size	1709
14.348virtio_block_discard_t Struct Reference	1709
14.348.1Detailed Description	1709
14.349virtio_block_header_t Struct Reference	1710
14.349.1Detailed Description	1710
14.350virtio_config_hdr_t Struct Reference	1711
14.350.1Detailed Description	1712
14.350.2Field Documentation	1712
14.350.2.1magic	1712
14.350.2.2status	1712
14.351virtio_config_queue_t Struct Reference	1713
14.351.1Detailed Description	1714

15 File Documentation	1715
15.1 amd64/l4/util/apic.h File Reference	1715
15.1.1 Detailed Description	1715
15.2 apic.h	1716
15.3 x86/l4/util/apic.h File Reference	1720
15.3.1 Detailed Description	1721
15.4 apic.h	1721
15.5 amd64/l4/util/idt.h File Reference	1726
15.5.1 Detailed Description	1727
15.6 idt.h	1727
15.7 x86/l4/util/idt.h File Reference	1728
15.7.1 Detailed Description	1728
15.8 idt.h	1729
15.9 amd64/l4/util/perform.h File Reference	1729
15.9.1 Detailed Description	1730
15.10perform.h	1730
15.11x86/l4/util/perform.h File Reference	1735
15.11.1 Detailed Description	1736
15.12perform.h	1736
15.13amd64/l4/util/rdtsc.h File Reference	1741
15.13.1 Detailed Description	1743
15.14rdtsc.h	1743
15.15x86/l4/util/rdtsc.h File Reference	1746
15.15.1 Detailed Description	1747
15.16rdtsc.h	1747
15.17amd64/l4/util/spin.h File Reference	1751
15.17.1 Detailed Description	1751
15.18spin.h	1751
15.19x86/l4/util/spin.h File Reference	1752
15.19.1 Detailed Description	1752

15.20spin.h	1752
15.21amd64/l4/util/util.h File Reference	1753
15.21.1 Detailed Description	1754
15.21.2 Function Documentation	1754
15.21.2.1 l4_sleep()	1754
15.21.2.2 l4util_micros2l4to()	1755
15.22util.h	1755
15.23x86/l4/util/util.h File Reference	1756
15.23.1 Detailed Description	1757
15.23.2 Function Documentation	1757
15.23.2.1 l4_sleep()	1757
15.23.2.2 l4util_micros2l4to()	1757
15.24util.h	1758
15.25amd64/l4f/l4/sys/segment.h File Reference	1758
15.25.1 Detailed Description	1759
15.25.2 Function Documentation	1759
15.25.2.1 fiasco_amd64_set_fs()	1759
15.25.2.2 fiasco_amd64_set_segment_base()	1760
15.26segment.h	1760
15.27amd64/l4/sys/segment.h File Reference	1761
15.27.1 Detailed Description	1763
15.27.2 Enumeration Type Documentation	1763
15.27.2.1 L4_task_ldt_x86_consts	1763
15.27.3 Function Documentation	1763
15.27.3.1 fiasco_amd64_segment_info()	1763
15.27.3.2 fiasco_amd64_set_fs()	1764
15.27.3.3 fiasco_amd64_set_segment_base()	1764
15.28segment.h	1765
15.29x86/l4f/l4/sys/segment.h File Reference	1766
15.29.1 Detailed Description	1767

15.30segment.h	1767
15.31x86/I4/sys/segment.h File Reference	1768
15.31.1 Detailed Description	1769
15.31.2 Enumeration Type Documentation	1769
15.31.2.1 L4_task_ldt_x86_consts	1769
15.32segment.h	1769
15.33amd64/I4f/I4/util/port_io.h File Reference	1770
15.33.1 Detailed Description	1771
15.34port_io.h	1771
15.35amd64/I4/util/port_io.h File Reference	1771
15.35.1 Detailed Description	1772
15.36port_io.h	1772
15.37x86/I4f/I4/util/port_io.h File Reference	1772
15.37.1 Detailed Description	1773
15.37.2 Function Documentation	1774
15.37.2.1 I4util_ioport_map()	1774
15.38port_io.h	1775
15.39x86/I4/util/port_io.h File Reference	1775
15.39.1 Detailed Description	1777
15.40port_io.h	1777
15.41amd64/I4f/I4/util/setjmp.h File Reference	1779
15.41.1 Detailed Description	1780
15.41.2 Function Documentation	1780
15.41.2.1 I4_thread_longjmp()	1780
15.41.2.2 I4_thread_setjmp()	1781
15.42setjmp.h	1781
15.43x86/I4f/I4/util/setjmp.h File Reference	1782
15.43.1 Detailed Description	1783
15.43.2 Function Documentation	1783
15.43.2.1 I4_thread_longjmp()	1783

15.43.2.2 l4_thread_setjmp()	1784
15.44setjmp.h	1784
15.45arm/l4/sys/linkage.h File Reference	1785
15.45.1 Detailed Description	1785
15.46linkage.h	1785
15.47amd64/l4/sys/linkage.h File Reference	1786
15.47.1 Detailed Description	1786
15.48linkage.h	1786
15.49x86/l4/sys/linkage.h File Reference	1786
15.49.1 Detailed Description	1787
15.50linkage.h	1787
15.51arm/l4/sys/mem_op.h File Reference	1787
15.51.1 Detailed Description	1788
15.52mem_op.h	1789
15.53arm/l4/sys/vm.h File Reference	1790
15.53.1 Detailed Description	1790
15.54vm.h	1790
15.55arm/l4/util/bitops_arch.h File Reference	1791
15.55.1 Detailed Description	1791
15.56bitops_arch.h	1792
15.57amd64/l4/util/bitops_arch.h File Reference	1792
15.57.1 Detailed Description	1792
15.58bitops_arch.h	1792
15.59x86/l4/util/bitops_arch.h File Reference	1795
15.59.1 Detailed Description	1796
15.60bitops_arch.h	1796
15.61arm/l4/util/cpu.h File Reference	1799
15.61.1 Detailed Description	1799
15.62cpu.h	1800
15.63amd64/l4/util/cpu.h File Reference	1800

15.63.1 Detailed Description	1801
15.64cpu.h	1801
15.65x86/i4/util/cpu.h File Reference	1802
15.65.1 Detailed Description	1803
15.66cpu.h	1803
15.67arm/i4/util/i4_macros.h File Reference	1804
15.67.1 Detailed Description	1804
15.68i4_macros.h	1804
15.69amd64/i4/util/i4_macros.h File Reference	1805
15.69.1 Detailed Description	1805
15.70i4_macros.h	1805
15.71i4/util/i4_macros.h File Reference	1805
15.71.1 Detailed Description	1806
15.72i4_macros.h	1806
15.73x86/i4/util/i4_macros.h File Reference	1806
15.73.1 Detailed Description	1806
15.74i4_macros.h	1807
15.75arm/i4/util/mbi_argv.h File Reference	1807
15.75.1 Detailed Description	1807
15.76mbi_argv.h	1808
15.77amd64/i4/util/mbi_argv.h File Reference	1808
15.77.1 Detailed Description	1808
15.78mbi_argv.h	1809
15.79x86/i4/util/mbi_argv.h File Reference	1809
15.79.1 Detailed Description	1810
15.80mbi_argv.h	1810
15.81arm/i4/util/stack_impl.h File Reference	1811
15.81.1 Detailed Description	1811
15.81.2 Function Documentation	1811
15.81.2.1 l4util_stack_get_sp()	1811

15.82	stack_impl.h	1812
15.83	amd64/l4/util/stack_impl.h File Reference	1812
15.83.1	Detailed Description	1812
15.83.2	Function Documentation	1812
15.83.2.1	l4util_stack_get_sp()	1812
15.84	stack_impl.h	1813
15.85	x86/l4/util/stack_impl.h File Reference	1813
15.85.1	Detailed Description	1813
15.85.2	Function Documentation	1814
15.85.2.1	l4util_stack_get_sp()	1814
15.86	stack_impl.h	1814
15.87	arm/l4f/l4/sys/syscall_defs.h File Reference	1814
15.87.1	Detailed Description	1815
15.88	syscall_defs.h	1815
15.89	contrib/libio-io/l4/io/io.h File Reference	1815
15.89.1	Function Documentation	1817
15.89.1.1	l4io_get_root_device()	1817
15.89.1.2	l4io_iterate_devices()	1817
15.89.1.3	l4io_request_all_ioports()	1818
15.89.1.4	l4io_request_icu()	1818
15.90	io.h	1818
15.91	l4/sys/types.h File Reference	1819
15.91.1	Detailed Description	1822
15.91.2	Function Documentation	1822
15.91.2.1	l4_capability_next()	1822
15.92	types.h	1822
15.93	l4/cxx/avl_map File Reference	1825
15.93.1	Detailed Description	1826
15.94	avl_map	1826
15.95	l4/cxx/avl_set File Reference	1828

15.95.1 Detailed Description	1829
15.96avl_set	1829
15.97l4/cxx/avl_tree File Reference	1832
15.97.1 Detailed Description	1834
15.98avl_tree	1834
15.99l4/cxx/basic_ostream File Reference	1838
15.99.1 Detailed Description	1839
15.100basic_ostream	1839
15.101l4/cxx/basic_vector.h File Reference	1842
15.101.1 Detailed Description	1843
15.102basic_vector.h	1843
15.103l4/cxx/bits/bst.h File Reference	1843
15.103.1 Detailed Description	1845
15.104bst.h	1845
15.105l4/cxx/bits/bst_base.h File Reference	1847
15.105.1 Detailed Description	1849
15.106bst_base.h	1849
15.107l4/cxx/bits/bst_iter.h File Reference	1850
15.107.1 Detailed Description	1851
15.108bst_iter.h	1852
15.109l4/cxx/exceptions File Reference	1853
15.109.1 Detailed Description	1855
15.110exceptions	1855
15.111l4/cxx/iostream File Reference	1858
15.111.1 Detailed Description	1859
15.112ostream	1859
15.113l4/cxx/ipc_helper File Reference	1859
15.113.1 Detailed Description	1860
15.114ipc_helper	1861
15.115l4/cxx/ipc_stream File Reference	1861

15.115.1 Detailed Description	1864
15.115.2 Function Documentation	1864
15.115.2.1 operator<<() [1/4]	1864
15.115.2.2 operator<<() [2/4]	1865
15.115.2.3 operator<<() [3/4]	1866
15.115.2.4 operator<<() [4/4]	1866
15.115.2.5 operator>>() [1/6]	1867
15.115.2.6 operator>>() [2/6]	1868
15.115.2.7 operator>>() [3/6]	1869
15.115.2.8 operator>>() [4/6]	1869
15.115.2.9 operator>>() [5/6]	1870
15.115.2.10 operator>>() [6/6]	1871
15.116 pc_stream	1872
15.117 /cxx/l4iostream File Reference	1882
15.117.1 Detailed Description	1882
15.118 l4iostream	1883
15.119 /cxx/l4types.h File Reference	1883
15.119.1 Detailed Description	1884
15.120 l4types.h	1884
15.121 /cxx/main_thread File Reference	1885
15.121.1 Detailed Description	1886
15.122 main_thread	1886
15.123 /cxx/pair File Reference	1886
15.123.1 Detailed Description	1887
15.124 pair	1888
15.125 /cxx/std_exc_io File Reference	1888
15.125.1 Detailed Description	1889
15.126 std_exc_io	1889
15.127 /cxx/string.h File Reference	1890
15.127.1 Detailed Description	1891

15.128	tring.h	1892
15.129	/irq/irq.h File Reference	1892
15.129.1	Detailed Description	1894
15.130	q.h	1894
15.131	arm/l4/util/irq.h File Reference	1895
15.131.1	Detailed Description	1895
15.132	q.h	1896
15.133	amd64/l4/util/irq.h File Reference	1896
15.133.1	Detailed Description	1897
15.133.2	Function Documentation	1897
15.133.2.1	l4util_irq_acknowledge()	1897
15.134	q.h	1898
15.135	s86/l4/util/irq.h File Reference	1899
15.135.1	Detailed Description	1899
15.135.2	Function Documentation	1900
15.135.2.1	l4util_irq_acknowledge()	1900
15.136	q.h	1900
15.137	/sys/irq.h File Reference	1901
15.137.1	Detailed Description	1903
15.138	q.h	1903
15.139	/libedid/edid.h File Reference	1906
15.140	edid.h	1907
15.141	/re/c/dataspace.h File Reference	1908
15.141.1	Detailed Description	1909
15.142	dataspace.h	1909
15.143	/re/c/debug.h File Reference	1910
15.143.1	Detailed Description	1911
15.144	debug.h	1911
15.145	/re/c/dma_space.h File Reference	1912
15.145.1	Detailed Description	1913

15.145.2	Typedef Documentation	1913
15.145.2.1	re_dma_space_dma_addr_t	1913
15.145.3	Enumeration Type Documentation	1913
15.145.3.1	re_dma_space_direction	1913
15.145.3.2	re_dma_space_space_attrbs	1914
15.146	dma_space.h	1914
15.147	re/c/event.h File Reference	1915
15.147.1	Detailed Description	1916
15.148	event.h	1917
15.149	re/event.h File Reference	1917
15.149.1	Detailed Description	1918
15.150	event.h	1918
15.151	re/c/log.h File Reference	1919
15.151.1	Detailed Description	1920
15.152	log.h	1920
15.153	re/c/mem_alloc.h File Reference	1921
15.153.1	Detailed Description	1922
15.154	mem_alloc.h	1923
15.155	re/c/namespace.h File Reference	1924
15.155.1	Detailed Description	1925
15.156	namespace.h	1925
15.157	re/c/rm.h File Reference	1926
15.157.1	Detailed Description	1927
15.158	rm.h	1927
15.159	re/c/util/cap_alloc.h File Reference	1930
15.159.1	Detailed Description	1930
15.160	cap_alloc.h	1931
15.161	re/c/util/kumem_alloc.h File Reference	1931
15.161.1	Detailed Description	1932
15.161.2	Function Documentation	1932

15.161.2.14re_util_kumem_alloc()	1932
15.162kumem_alloc.h	1933
15.163/re/c/util/video/goos_fb.h File Reference	1933
15.163.1Detailed Description	1934
15.164goos_fb.h	1934
15.165/re/c/video/colors.h File Reference	1935
15.165.1Detailed Description	1936
15.166colors.h	1937
15.167/re/c/video/goos.h File Reference	1937
15.167.1Detailed Description	1939
15.168goos.h	1939
15.169/re/c/video/view.h File Reference	1940
15.169.1Detailed Description	1942
15.170view.h	1943
15.171/re/cap_alloc File Reference	1944
15.171.1Detailed Description	1945
15.172cap_alloc	1945
15.173/re/util/cap_alloc File Reference	1947
15.173.1Detailed Description	1949
15.174cap_alloc	1949
15.175/re/consts File Reference	1951
15.175.1Detailed Description	1952
15.176consts	1952
15.177/re/dataspace File Reference	1953
15.177.1Detailed Description	1954
15.178dataspace	1954
15.179/re/dataspace-sys.h File Reference	1955
15.179.1Detailed Description	1956
15.180dataspace-sys.h	1956
15.181/re/debug File Reference	1956

15.181.1 Detailed Description	1957
15.182 debug	1958
15.183/re/dma_space File Reference	1958
15.184 dma_space	1960
15.185/re/elf_aux.h File Reference	1961
15.185.1 Detailed Description	1962
15.186 elf_aux.h	1962
15.187/re/env File Reference	1963
15.187.1 Detailed Description	1964
15.188 env	1964
15.189/re/env.h File Reference	1965
15.189.1 Detailed Description	1967
15.189.2 Typedef Documentation	1967
15.189.2.1 re_env_t	1967
15.190 env.h	1967
15.191/re/error_helper File Reference	1969
15.191.1 Detailed Description	1971
15.192 error_helper	1971
15.193/re/util/event File Reference	1972
15.194 event	1973
15.195/re/impl/dataspace_impl.h File Reference	1975
15.195.1 Detailed Description	1976
15.196 dataspace_impl.h	1976
15.197/re/impl/mem_alloc_impl.h File Reference	1977
15.197.1 Detailed Description	1978
15.198 mem_alloc_impl.h	1978
15.199/re/impl/namespace_impl.h File Reference	1979
15.199.1 Detailed Description	1980
15.200 namespace_impl.h	1980
15.201/re/impl/rm_impl.h File Reference	1981

15.201. Detailed Description	1982
15.202m_impl.h	1982
15.203re/l4aux.h File Reference	1984
15.203. Detailed Description	1984
15.204l4aux.h	1985
15.205re/log File Reference	1985
15.205. Detailed Description	1987
15.206log	1987
15.207re/log-sys.h File Reference	1987
15.207. Detailed Description	1988
15.208log-sys.h	1988
15.209re/mem_alloc File Reference	1988
15.209. Detailed Description	1990
15.210mem_alloc	1990
15.211re/mem_alloc-sys.h File Reference	1990
15.211. Detailed Description	1991
15.212mem_alloc-sys.h	1991
15.213re/namespace File Reference	1992
15.213. Detailed Description	1993
15.214namespace	1993
15.215re/namespace-sys.h File Reference	1994
15.215. Detailed Description	1995
15.216namespace-sys.h	1995
15.217re/parent File Reference	1995
15.217. Detailed Description	1997
15.218parent	1997
15.219re/parent-sys.h File Reference	1997
15.219. Detailed Description	1998
15.220parent-sys.h	1998
15.221re/protocols.h File Reference	1998

15.221.1 Detailed Description	1999
15.221.2 Enumeration Type Documentation	1999
15.221.2.1 L4re_protocols	1999
15.222 protocols.h	1999
15.223 /re/rm File Reference	2000
15.223.1 Detailed Description	2001
15.224 m	2001
15.225 /re/rm-sys.h File Reference	2005
15.225.1 Detailed Description	2006
15.226 m-sys.h	2006
15.227 /re/shared_cap File Reference	2006
15.227.1 Detailed Description	2008
15.228 shared_cap	2008
15.229 /re/util/shared_cap File Reference	2009
15.229.1 Detailed Description	2010
15.230 shared_cap	2010
15.231 /re/unique_cap File Reference	2011
15.231.1 Detailed Description	2013
15.232 unique_cap	2013
15.233 /re/util/unique_cap File Reference	2013
15.233.1 Detailed Description	2015
15.234 unique_cap	2015
15.235 /re/util/bitmap_cap_alloc File Reference	2016
15.235.1 Detailed Description	2017
15.236 bitmap_cap_alloc	2017
15.237 /re/util/cap File Reference	2018
15.237.1 Detailed Description	2019
15.238 ap	2020
15.239 /re/util/cap_alloc_impl.h File Reference	2020
15.239.1 Detailed Description	2021

15.240	cap_alloc_impl.h	2021
15.241	re/util/counting_cap_alloc File Reference	2022
15.241.1	Detailed Description	2023
15.242	counting_cap_alloc	2023
15.243	re/util/item_alloc File Reference	2025
15.243.1	Detailed Description	2026
15.244	kmem_alloc	2026
15.245	re/util/kmem_alloc File Reference	2027
15.245.1	Detailed Description	2028
15.246	kumem_alloc	2028
15.247	re/util/region_mapping File Reference	2029
15.247.1	Detailed Description	2030
15.248	region_mapping	2030
15.249	re/video/goos-sys.h File Reference	2035
15.249.1	Detailed Description	2036
15.250	goos-sys.h	2036
15.251	shmc/shmc.h File Reference	2036
15.251.1	Detailed Description	2038
15.252	shm.h	2039
15.253	sigma0/sigma0.h File Reference	2041
15.253.1	Detailed Description	2043
15.254	sigma0.h	2043
15.255	sys/__kernel_object_impl.h File Reference	2044
15.255.1	Detailed Description	2045
15.256	_kernel_object_impl.h	2045
15.257	sys/__ktrace-impl.h File Reference	2046
15.257.1	Detailed Description	2047
15.258	_ktrace-impl.h	2047
15.259	sys/__typeinfo.h File Reference	2048
15.259.1	Detailed Description	2051

15.260	_typeinfo.h	2051
15.261	arm/sys/cache.h File Reference	2061
15.261.1	Detailed Description	2062
15.262	cache.h	2062
15.263	arm/l4/sys/cache.h File Reference	2063
15.263.1	Detailed Description	2064
15.264	cache.h	2064
15.265	amd64/l4/sys/cache.h File Reference	2066
15.265.1	Detailed Description	2066
15.266	cache.h	2066
15.267	x86/l4/sys/cache.h File Reference	2067
15.267.1	Detailed Description	2067
15.268	cache.h	2068
15.269	l4/sys/capability File Reference	2068
15.269.1	Detailed Description	2070
15.269.2	Macro Definition Documentation	2070
15.269.2.1	L4_DISABLE_COPY	2070
15.270	capability	2070
15.271	l4/sys/compiler.h File Reference	2072
15.271.1	Detailed Description	2073
15.272	compiler.h	2073
15.273	l4/sys/consts.h File Reference	2075
15.273.1	Detailed Description	2077
15.274	consts.h	2077
15.275	arm/l4/sys/consts.h File Reference	2079
15.275.1	Detailed Description	2079
15.276	consts.h	2080
15.277	amd64/l4/sys/consts.h File Reference	2080
15.277.1	Detailed Description	2080
15.278	consts.h	2081

15.279	14/sys/consts.h File Reference	2081
15.279.1	Detailed Description	2081
15.280	14/consts.h	2082
15.281	14/re/consts.h File Reference	2082
15.281.1	Detailed Description	2083
15.282	20/consts.h	2083
15.283	24/sys/cxx/ipc_client File Reference	2083
15.283.1	Macro Definition Documentation	2085
15.283.1.1	L4_RPC_DEF	2085
15.284	24/cpc_client	2085
15.285	24/sys/cxx/ipc_iface File Reference	2086
15.285.1	Detailed Description	2088
15.285.2	Macro Definition Documentation	2088
15.285.2.1	L4_INLINE_RPC	2088
15.285.2.2	L4_INLINE_RPC_NF	2089
15.285.2.3	L4_INLINE_RPC_NF_OP	2089
15.285.2.4	L4_INLINE_RPC_OP	2090
15.285.2.5	L4_RPC	2091
15.285.2.6	L4_RPC_NF	2091
15.285.2.7	L4_RPC_NF_OP	2092
15.285.2.8	L4_RPC_OP	2092
15.286	24/cpc_iface	2093
15.287	24/cxx/ipc_server File Reference	2098
15.287.1	Detailed Description	2099
15.288	24/cpc_server	2099
15.289	24/sys/cxx/ipc_types File Reference	2100
15.290	24/cpc_types	2102
15.291	24/sys/cxx/smart_capability_1x File Reference	2109
15.292	24/smart_capability_1x	2110
15.293	24/sys/cxx/types File Reference	2113

15.294	types	2114
15.295	/sys/debugger File Reference	2115
15.295.1	Detailed Description	2116
15.296	debugger	2116
15.297	/sys/debugger.h File Reference	2117
15.297.1	Detailed Description	2119
15.297.2	Function Documentation	2119
15.297.2.1	4_debugger_get_object_name()	2119
15.297.2.2	4_debugger_query_log_name()	2119
15.297.2.3	4_debugger_query_log_typeid()	2120
15.297.2.4	4_debugger_switch_log()	2121
15.298	debugger.h	2121
15.299	/sys/err.h File Reference	2124
15.299.1	Detailed Description	2125
15.300	err.h	2126
15.301	/sys/exception File Reference	2126
15.301.1	Detailed Description	2127
15.302	exception	2128
15.303	/sys/factory File Reference	2128
15.303.1	Detailed Description	2130
15.304	factory	2130
15.305	/sys/factory.h File Reference	2132
15.305.1	Detailed Description	2133
15.306	factory.h	2133
15.307	/sys/icu File Reference	2138
15.307.1	Detailed Description	2139
15.308	icu	2139
15.309	/sys/icu.h File Reference	2139
15.309.1	Detailed Description	2142
15.310	icu.h	2142

15.3114/sys/ipc.h File Reference	2145
15.311.1 Detailed Description	2147
15.311.2 Function Documentation	2147
15.311.2.1 l4_ipc_to_errno()	2147
15.311pc.h	2148
15.311arm/l4f/l4/sys/ipc.h File Reference	2151
15.313.1 Detailed Description	2152
15.311pc.h	2152
15.31186/l4f/l4/sys/ipc.h File Reference	2153
15.315.1 Detailed Description	2154
15.311pc.h	2154
15.3117/sys/ipc_gate File Reference	2154
15.317.1 Detailed Description	2156
15.311pc_gate	2156
15.3119/sys/ipc_gate.h File Reference	2156
15.319.1 Detailed Description	2158
15.320pc_gate.h	2158
15.3214/sys/irq File Reference	2159
15.321.1 Detailed Description	2161
15.322q	2161
15.323/sys/kernel_object.h File Reference	2163
15.323.1 Detailed Description	2164
15.324kernel_object.h	2164
15.3254/sys/kip File Reference	2165
15.325.1 Detailed Description	2166
15.326ip	2167
15.3274/sys/ktrace.h File Reference	2168
15.327.1 Detailed Description	2170
15.328trace.h	2170
15.3294/sys/l4int.h File Reference	2171

15.329. Detailed Description	2172
15.330.int.h	2172
15.331.arm/l4/sys/l4int.h File Reference	2173
15.331. Detailed Description	2173
15.332.int.h	2173
15.333.amd64/l4/sys/l4int.h File Reference	2173
15.333. Detailed Description	2174
15.334.int.h	2174
15.335.x86/l4/sys/l4int.h File Reference	2174
15.335. Detailed Description	2175
15.336.int.h	2175
15.337./sys/memdesc.h File Reference	2175
15.337. Detailed Description	2177
15.338.memdesc.h	2177
15.339./sys/meta File Reference	2179
15.339. Detailed Description	2181
15.340.meta	2181
15.341./sys/pager File Reference	2181
15.341. Detailed Description	2183
15.342.pager	2183
15.343./sys/platform_control File Reference	2183
15.343. Detailed Description	2184
15.344.platform_control	2185
15.345./sys/platform_control.h File Reference	2185
15.345. Detailed Description	2187
15.346.platform_control.h	2187
15.347./sys/rcv_endpoint File Reference	2189
15.347. Detailed Description	2190
15.348.rcv_endpoint	2190
15.349./sys/rcv_endpoint.h File Reference	2191

15.349.1 Detailed Description	2192
15.349.2 Enumeration Type Documentation	2192
15.349.2.1 L4_rcv_ep_ops	2192
15.350 cv_endpoint.h	2193
15.351 /sys/scheduler File Reference	2193
15.351.1 Detailed Description	2194
15.352 scheduler	2195
15.353 /sys/scheduler.h File Reference	2195
15.353.1 Detailed Description	2197
15.354 scheduler.h	2197
15.355 /sys/semaphore File Reference	2200
15.355.1 Detailed Description	2202
15.356 semaphore	2202
15.357 /sys/smart_capability File Reference	2202
15.357.1 Detailed Description	2204
15.358 smart_capability	2204
15.359 /sys/task File Reference	2206
15.359.1 Detailed Description	2208
15.360 task	2208
15.361 /sys/task.h File Reference	2209
15.361.1 Detailed Description	2210
15.362 task.h	2210
15.363 /sys/thread File Reference	2214
15.363.1 Detailed Description	2215
15.364 thread	2215
15.365 /cxx/thread File Reference	2217
15.365.1 Detailed Description	2218
15.366 thread	2218
15.367 /sys/typeinfo_svr File Reference	2219
15.367.1 Detailed Description	2221

15.368	ypinfo_svr	2221
15.369	/sys/utcb.h File Reference	2221
15.369.1	Detailed Description	2223
15.370	utcb.h	2223
15.371	arm/l4/sys/utcb.h File Reference	2226
15.371.1	Detailed Description	2227
15.372	utcb.h	2227
15.373	amd64/l4/sys/utcb.h File Reference	2228
15.373.1	Detailed Description	2230
15.374	utcb.h	2230
15.375	86/l4/sys/utcb.h File Reference	2231
15.375.1	Detailed Description	2233
15.376	utcb.h	2233
15.377	/sys/vcon File Reference	2234
15.377.1	Detailed Description	2236
15.378	con	2236
15.379	/sys/vcon.h File Reference	2236
15.379.1	Detailed Description	2238
15.379.2	Enumeration Type Documentation	2239
15.379.2.1	L4_vcon_read_flags	2239
15.380	con.h	2239
15.381	/sys/vhw.h File Reference	2242
15.381.1	Detailed Description	2243
15.382	hw.h	2244
15.383	/sys/vm File Reference	2245
15.383.1	Detailed Description	2246
15.384	m	2246
15.385	/util/alloc.h File Reference	2246
15.385.1	Detailed Description	2247
15.386	alloc.h	2247

15.387/cxx/alloc.h File Reference	2248
15.387.1 Detailed Description	2248
15.388/alloc.h	2248
15.389/util/assert.h File Reference	2249
15.389.1 Detailed Description	2250
15.390/assert.h	2250
15.391/4/sys/assert.h File Reference	2251
15.391.1 Detailed Description	2252
15.391.2 Macro Definition Documentation	2253
15.391.2.1 4_assert	2253
15.392/assert.h	2253
15.393/4/util/atomic.h File Reference	2254
15.393.1 Detailed Description	2256
15.394/atomic.h	2257
15.395/arm/4/sys/atomic.h File Reference	2261
15.395.1 Detailed Description	2261
15.396/atomic.h	2262
15.397/cxx/atomic.h File Reference	2262
15.397.1 Detailed Description	2263
15.398/atomic.h	2263
15.399/4/util/backtrace.h File Reference	2263
15.399.1 Detailed Description	2264
15.399.2 Function Documentation	2264
15.399.2.1 4util_backtrace()	2265
15.400/backtrace.h	2265
15.401/4/util/base64.h File Reference	2265
15.401.1 Detailed Description	2266
15.402/base64.h	2267
15.403/4/util/bitops.h File Reference	2267
15.403.1 Detailed Description	2269

15.404	bits.h	2269
15.405	util/elf.h File Reference	2272
15.405.1	Detailed Description	2285
15.406	elf.h	2285
15.407	util/getopt.h File Reference	2295
15.407.1	Detailed Description	2295
15.408	getopt.h	2295
15.409	util/keymap.h File Reference	2296
15.409.1	Detailed Description	2297
15.410	keymap.h	2297
15.411	util/kip.h File Reference	2297
15.412	kip.h	2298
15.413	sys/kip.h File Reference	2299
15.413.1	Detailed Description	2300
15.414	kip.h	2300
15.415	util/kprintf.h File Reference	2302
15.415.1	Detailed Description	2302
15.416	kprintf.h	2303
15.417	util/list_alloc.h File Reference	2303
15.417.1	Detailed Description	2304
15.417.2	Function Documentation	2304
15.417.2.1	la_alloc()	2304
15.417.2.2	la_avail()	2304
15.417.2.3	la_dump()	2305
15.417.2.4	la_free()	2305
15.417.2.5	la_init()	2305
15.418	st_alloc.h	2306
15.419	util/lock.h File Reference	2306
15.419.1	Detailed Description	2307
15.420	lock.h	2307

15.4214/	util/mb_info.h File Reference	2308
15.421.1	Detailed Description	2310
15.421.2	Macro Definition Documentation	2310
15.421.2.1	L4UTIL_MB_MEMORY	2310
15.421.2.2	MB_ARD_MEMORY	2311
15.421.2.3	MB_ART_MEMORY	2311
15.421	mb_info.h	2311
15.42314/	util/parse_cmd.h File Reference	2315
15.423.1	Detailed Description	2315
15.423	parse_cmd.h	2316
15.42514/	util/rand.h File Reference	2316
15.425.1	Detailed Description	2317
15.425	rand.h	2317
15.42714/	util/reboot.h File Reference	2318
15.427.1	Detailed Description	2318
15.427	reboot.h	2318
15.42914/	util/sll.h File Reference	2319
15.429.1	Detailed Description	2319
15.43014/	util/sll.h	2319
15.43114/	util/splitlog2.h File Reference	2323
15.431.1	Detailed Description	2323
15.431	splitlog2.h	2324
15.43314/	util/stack.h File Reference	2324
15.433.1	Detailed Description	2325
15.433.2	Function Documentation	2325
15.433.2.1	util_stack_get_sp()	2326
15.433	stack.h	2326
15.43514/	util/thread.h File Reference	2326
15.435.1	Detailed Description	2327
15.435.2	Macro Definition Documentation	2328

15.435.2.1 __L4UTIL_THREAD_FUNC	2328
15.436 thread.h	2328
15.437 /sys/thread.h File Reference	2329
15.437.1 Detailed Description	2331
15.438 thread.h	2331
15.439 arm/l4/sys/thread.h File Reference	2339
15.439.1 Detailed Description	2339
15.440 thread.h	2339
15.441 /vbus/vbus.h File Reference	2340
15.441.1 Detailed Description	2341
15.441.2 Enumeration Type Documentation	2341
15.441.2.1 anonymous enum	2341
15.442 bus.h	2342
15.443 /vbus/vbus_interfaces.h File Reference	2343
15.443.1 Detailed Description	2344
15.443.2 Enumeration Type Documentation	2344
15.443.2.1 anonymous enum	2344
15.443.2.2 vbus_iface_type_t	2344
15.443.3 Function Documentation	2344
15.443.3.1 l4vbus_subinterface_supported()	2344
15.444 bus_interfaces.h	2345
15.445 /vbus/vbus_types.h File Reference	2345
15.445.1 Detailed Description	2347
15.445.2 Enumeration Type Documentation	2347
15.445.2.1 l4vbus_device_flags_t	2347
15.445.2.2 vbus_resource_type_t	2347
15.446 bus_types.h	2347

16 Example Documentation	2349
16.1 examples/clntsrv/client.cc	2349
16.2 examples/clntsrv/clntsrv.cfg	2350
16.3 examples/clntsrv/server.cc	2350
16.4 examples/libs/l4re/c++/mem_alloc/ma+rm.cc	2351
16.5 examples/libs/l4re/c++/shared_ds/ds_clnt.cc	2352
16.6 examples/libs/l4re/c++/shared_ds/ds_srv.cc	2353
16.7 examples/libs/l4re/c++/shared_ds/shared_ds.cfg	2355
16.8 examples/libs/l4re/c/ma+rm.c	2356
16.9 examples/libs/l4re/streammap/client.cc	2357
16.10examples/libs/l4re/streammap/server.cc	2358
16.11examples/libs/l4re/streammap/streammap.cfg	2359
16.12examples/libs/libirq/async_isr.c	2359
16.13examples/libs/libirq/loop.c	2360
16.14examples/libs/shmc/prodcons.c	2361
16.15examples/sys/aliens/main.c	2362
16.16examples/sys/ipc/ipc.cfg	2366
16.17examples/sys/ipc/ipc_example.c	2366
16.18examples/sys/isr/main.c	2368
16.19examples/sys/migrate/thread_migrate.cc	2369
16.20examples/sys/migrate/thread_migrate.cfg	2370
16.21examples/sys/singlestep/main.c	2371
16.22examples/sys/start-with-exc/main.c	2373
16.23examples/sys/utcb-ipc/main.c	2375
16.24examples/sys/ux-vhw/main.c	2376
16.25hello/server/src/main.c	2377
16.26tmpfs/lib/src/fs.cc	2377
Index	2385

Chapter 1

Overview

Welcome to the documentation of the L4 Runtime Environment, or L4Re for short. There are two parts to this documentation: a user manual, which provides a birds eye view of L4Re and its environment, and a reference section which documents the complete programming API.

User Manual

1. [Introduction](#) shortly explains the concept of microkernels and introduces the basic terminology.
2. [Tutorial](#) helps you getting started with setting up the development environment and writing your own first L4Re application.
3. [Programming for L4Re](#) explains in detail the most important programming concepts.
4. [L4Re Servers](#) provides a quick overview over standard services running on the L4Re operating system.

Reference

The second part provides the complete reference of all classes and functions of the L4Re environment as well as a list of example code.

Chapter 2

Introduction

The intention of this section is to provide a short overview about the [L4Re](#) environment.

The general structure of a microkernel-based system will be introduced and the principal functionality of the servers in the basic environment outlined.

2.1 Fiasco.OC

The Fiasco.OC microkernel is the lowest-level piece of software running in an L4-based system. The microkernel is the only program that runs in privileged processor mode. It does not include complex services such as program loading, device drivers, or file systems; those are implemented in user-level programs on top of it (a basic set of these services and abstractions is provided by the [L4 Runtime Environment](#)).

Fiasco.OC kernel services are implemented in kernel objects. Tasks hold references to kernel objects in their respective *"object space"*, which is a kernel-protected table. These references are called *capabilities*. Fiasco system calls are function invocations on kernel objects through the corresponding capabilities. These can be thought of as function invocations on object references in an object-oriented programming environment. Furthermore, if a task owns a capability, it may grant other tasks the same (or fewer) rights on this object by passing the capability from its own to the other task's object space.

From a design perspective, capabilities are a concept that enables flexibility in the system structure. A thread that invokes an object through a capability does not need to care about where this object is implemented. In fact, it is possible to implement all objects either in the kernel or in a user-level server and replace one implementation with the other transparently for clients.

2.1.1 Communication

The basic communication mechanism in L4-based systems is called *"Inter Process Communication (IPC)"*. It is always synchronous, i.e. both communication partners need to actively rendezvous for IPC. In addition to transmitting arbitrary data between threads, IPC is also used to resolve hardware exceptions, faults and for virtual memory management.

2.1.2 Kernel Objects

The following list gives a short overview of the kernel objects provided by the Fiasco.OC microkernel:

- **Task** A task comprises a memory address space (represented by the task's page table), an object space (holding the kernel protected capabilities), and on x86 an IO-port address space.
- **Thread** A thread is bound to a task and executes code. Multiple threads can coexist in one task and are scheduled by the Fiasco scheduler.
- **Factory** A factory is used by applications to create new kernel objects. Access to a factory is required to create any new kernel object. Factories can control and restrict object creation.
- **IPC Gate** An IPC gate is used to create a secure communication channel between different tasks. It embeds a label (kernel protected payload) that securely identifies the gate through which a message is received. The gate label is not visible to and cannot be altered by the sender.
- **IRQ** IRQ objects provide access to hardware interrupts. Additionally, programs can create new virtual interrupt objects and trigger them. This allows to implement a signaling mechanism. The receiver cannot decide whether the interrupt is a physical or virtual one.
- **Vcon** Provides access to the in-kernel debugging console (input and output). There is only one such object in the kernel and it is only available, if the kernel is built with debugging enabled. This object is typically interposed through a user-level service or without debugging in the kernel can be completely based on user-level services.
- **Scheduler** Implements scheduling policy and assignment of threads to CPUs, including CPU statistics.

2.2 L4Re System Structure

The system has a multi-tier architecture consisting of the following layers depicted in the figure below:

- **Microkernel** The microkernel is the component at the lowest level of the software stack. It is the only piece of software that is running in the privileged mode of the processor.
- **Tasks** Tasks are the basic containers (address spaces) in which system services and applications are executed. They run in the processor's deprivileged user mode.

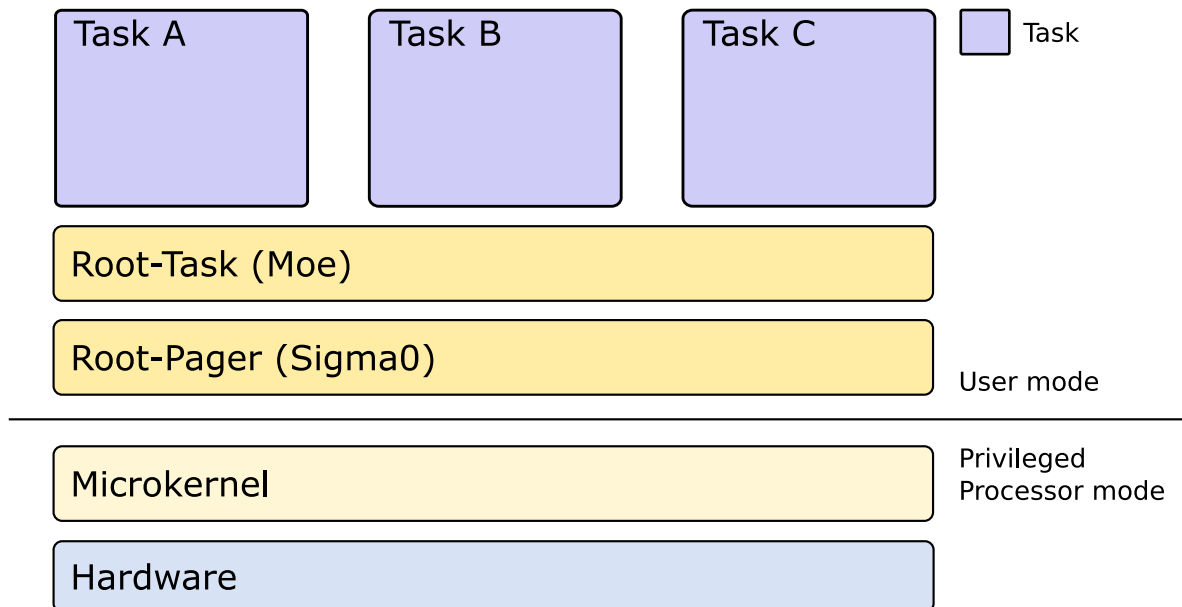


Figure 2.1 Basic Structure of an L4Re based system

In terms of functionality, the system is structured as follows:

- **Microkernel** The kernel provides primitives to execute programs in tasks, to enforce isolation among them, and to provide means of secure communication in order to let them cooperate. As the kernel is the most privileged, security-critical software component in the system, it is a general design goal to make it as small as possible in order to reduce its attack surface. It provides only a minimal set of mechanisms that are necessary to support applications.
- **Runtime Environment** The small kernel offers a concise set of interfaces, but these are not necessarily suited for building applications directly on top of it. The [L4](#) Runtime Environment aims at providing more convenient abstractions for application development. It comprises low-level software components that interface directly with the microkernel. The root pager *sigma0* and the root task *Moe* are the most basic components of the runtime environment. Other services (e.g., for device enumeration) use interfaces provided by them.
- **Applications** Applications run on top of the system and use services provided by the Runtime Environment – or by other applications. There may be several types of applications in the system and even virtual machine monitors and device drivers are considered applications in the terminology used in this document. They are running alongside other applications on the system.

Lending terminology from the distributed systems area, applications offering services to other applications are usually called *servers*, whereas applications using those services are named *clients*. Being in both roles is also common, for instance, a file system server may be viewed as a server with respect to clients using the file system, while the server itself may also act as a client of a hard disk driver.

2.3 L4 Runtime Environment (L4Re)

The L4 Runtime Environment (L4Re) provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the Fiasco.OC microkernel.

L4Re consists of a set of libraries and servers. Libraries as well as server interfaces are completely object oriented. They implement prototype implementations for the classes defined by the L4Re specification.

A minimal L4Re-based application needs 3 components to be booted beforehand: the Fiasco microkernel, the root pager (Sigma0), and the root task (Moe). The Sigma0 root pager initially owns all system resources, but is usually used only to resolve page faults for the Moe root task. Moe provides the essential services to normal user applications such as an initial program loader, a region-map service for virtual memory management, and a memory (data space) allocator.

Chapter 3

Tutorial

This tutorial assumes that the reader is familiar with the basic L4 concepts that were discussed in the [Introduction](#) section and has a working knowledge of C++.

Here you can find the first steps to boot a very simple setup. The setup consists of the following components:

- Fiasco.OC — Microkernel
- Sigma0 — Root Pager
- Moe — Root Task
- Ned — Init Process
- hello — Hello World Application

The guide assumes that you already compiled the base components and describes how to generate an ISO image, with GRUB 1 or GRUB 2 as a boot loader, that can for example be booted within QEMU.

First you need a `modules.list` file that contains an entry for the scenario.

```
modaddr 0x002000000

entry hello
  kernel fiasco -serial_esc
  roottask moe rom/hello.cfg
  module l4re
  module ned
  module hello.cfg
  module hello
```

This file describes all the binaries and scripts to put into the ISO image, and also describes the GRUB `menu.lst` contents. What you need to do is to set the `make` variable `MODULE_SEARCH_PATH` to contain the path to your Fiasco.OC build directory and the directory containing your `hello.cfg` script.

The `hello.cfg` script should look like the following. A ready to use version can be found in `I4/conf/examples`.

```
local L4 = require("L4");
L4.default_loader:start({}, "rom/hello");
```

The first line of this script ensures that the [L4](#) package is available for the script. The second line uses the default loader object defined in that package and starts the binary `rom/hello`.

Note

All modules defined in `modules.list` are available as data spaces ([L4Re::Dataspace](#)) and registered in a name space ([L4Re::Namespace](#)). This name space is in turn available as 'rom' to the init process ([Ned](#)).

Now you can go to your [L4Re](#) build directory and run the following command.

Note

The example assumes that you have created the `modules.list` and `hello.cfg` files in the `/tmp` directory. Adapt if you created them somewhere else.

```
make grubliso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Or as an alternative use GRUB 2:

```
make grub2iso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Now you should be able to boot the image in QEMU by running:

```
qemu-system-i386 -cdrom images/hello.iso -serial stdio
```

If you press `<ESC>` in the terminal that shows you the serial output you enter the Fiasco.OC kernel debugger... Have fun.

3.1 Customizations

A basic set of bootable entries can be found in `l4/conf/modules.list`. This file is the default for any image creation as shown above. It is recommended that local modification regarding image creation are done in `conf/Makeconf.boot`. Initially you may copy `Makeconf.boot.example` to `Makeconf.boot`. You can overwrite `MODULES_LIST` to set your own modules-list file. Set `MODULE_SEARCH_PATH` to your setup according to the examples given in the file. When configured a `make` call is reduced to:

```
make grub2iso E=hello
```

All other local configuration can be done in a `Makeconf.local` file located in the `l4` directory.

Chapter 4

Programming for L4Re

This part of the documentation discusses the concept of microkernel-based programming in more detail.

You should have already a basic understanding of the [L4Re](#) programming environment from the tutorial.

- [Capabilities and Naming](#)
- [Initial Environment and Application Bootstrapping](#)
- [Memory management - Data Spaces and the Region Map](#)
- [Program Input and Output](#)
- [Initial Memory Allocator and Factory](#)
- [Application and Server Building Blocks](#)
- [Pthread Support](#)
- tasks and threads
- communication channels
- server loops
- [Interface Definition Language](#)
- hardware access
- [L4Re Build System](#)

4.1 Capabilities and Naming

The [L4Re](#) system is a capability based system which uses and offers capabilities to implement fine-grained access control.

Generally, owning a capability means to be allowed to communicate with the object the capability points to. All user-visible kernel objects, such as tasks, threads, and IRQs, can be accessed only through a capability. Please refer to the [Kernel Objects](#) documentation for details. Capabilities are stored in per-task capability tables (the object space) and are referenced by capability selectors or object flex pages. In a simplified view, a capability selector is a natural number indexing into the capability table of the current task.

As a matter of fact, a system designed solely based on capabilities, uses so-called 'local names', because each task can only access those objects made available to this task. Other objects are not visible to and accessible by the task.

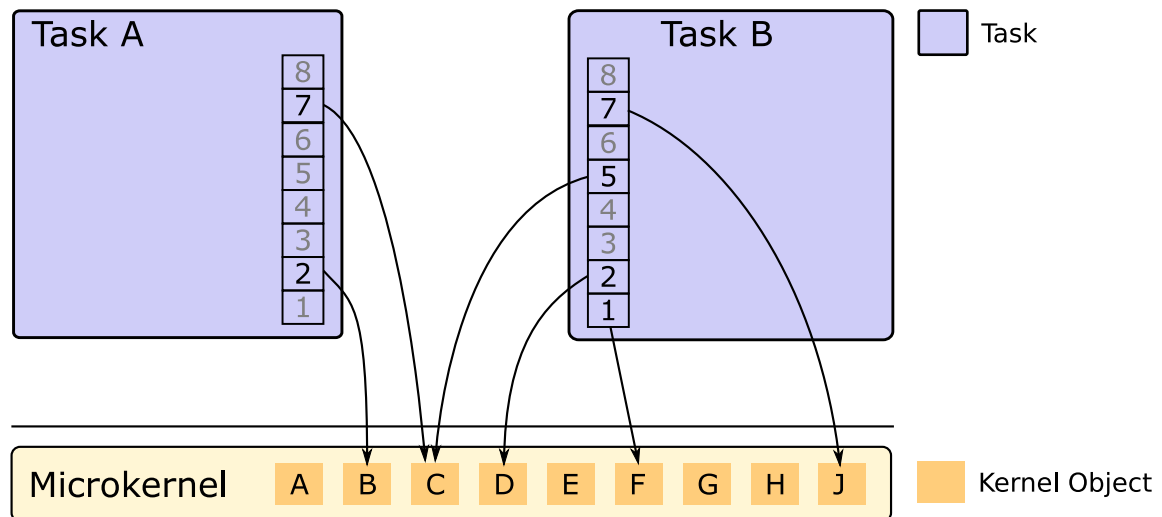


Figure 4.1 Capabilities and Local Naming in L4

So how does an application get access to a service? In general all applications are started with an initial set of objects available. This set of objects is predetermined by the creator of a new application process and granted directly to the new task before starting the first application thread. The application can then use these initial objects to request access to further objects or to transfer capabilities to own objects to other applications. A central L4Re object for exchanging capabilities at runtime is the name-space object, implementing a store of named capabilities.

From a security perspective, the set of initial capabilities (access rights to objects) completely define the execution environment of an application. Mandatory security policies can be defined by well known properties of the initial objects and carefully handled access rights to them.

4.2 Initial Environment and Application Bootstrapping

New applications that are started by a loader conforming to L4Re get provided an Initial Environment.

This environment comprises a set of capabilities to initial L4Re objects that are required to bootstrap and run this application. These capabilities include:

- A capability to an initial memory allocator for obtaining memory in the form of data spaces

- A capability to a factory which can be used to create additional kernel objects
- A capability to a Vcon object for debugging output and maybe input
- A set of named capabilities to application specific objects

During the bootstrapping of the application, the loader establishes data spaces for each individual region in the ELF binary. These include data spaces for the code and data sections, and a data space backed with RAM for the stack of the program's first thread.

One loader implementation is the `moe` root task. Moe usually starts an *init* process that is responsible for coordinating the further boot process. The default *init* process is `ned`, which implements a script-based configuration and startup of other processes. Ned uses Lua (<http://www.lua.org>) as its scripting language, see [Ned Script example](#) for more details.

4.2.1 Configuring an application before startup

The default [L4Re](#) init process (Ned) provides a Lua script based configuration of initial capabilities and application startup. Ned itself also has a set of initial objects available that can be used to create the environment for an application. The most important object is a kernel object factory that allows creation of kernel objects such as IPC gates (communication channels), tasks, threads, etc. Ned uses Lua tables (associative arrays) to represent sets of capabilities that shall be granted to application processes.

```
local caps = {
    name = some_capability
}
```

The [L4](#) Lua package in Ned also has support functions to create application tasks, region-map objects, etc. to start an ELF binary in a new task. The package also contains Lua bindings for basic [L4Re](#) objects, for example, to generic factory objects, which are used to create kernel objects and also user-level objects provided by user-level servers.

```
L4.default_loader:start({ caps = { some_service = service } }, "rom/program --arg");
```

4.2.2 Connecting clients and servers

In general, a connection between a client and a server is represented by a communication channel (IPC gate). That is available to the client and the server. You can see the simplest connection between a client and a server in the following example.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel(); -- create an IPC gate
loader:start({ caps = { service = svc:svr() } }, "rom/my_server");
loader:start({ caps = { service = svc:m("rw") } }, "rom/my_client");
```

As you can see in the snippet, the first action is to create a new channel (IPC gate) using `loader:new_channel()`. The capability to the gate is stored in the variable `svc`. Then the binary `my_server` is started in a new task, and full (`:svr()`) access to the IPC gate is granted to the server as initial object. The gate is accessible to the server application as "service" in the set of its initial capabilities. Virtually in parallel a second task, running the client application, is started and also given access to the IPC gate with less rights (`:m("rw")`), note, this is essential). The server can now receive messages via the IPC gate and provide some service and the client can call operations on the IPC gate to communicate with the server.

Services that keep client specific state need to implement per-client server objects. Usually it is the responsibility of some authority (e.g., Ned) to request such an object from the service via a generic factory object that the service provides initially.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel():m("rws"); -- create an IPC gate with rws rights
loader:start({ caps = { service = svc:svr() } }, "rom/my-service");
loader:start({ caps = { foo_service = svc:create(object_to_create, "param") }}, "rom/client");
```

This example is quite similar to the first one, however, the difference is that Ned itself calls the `create` method on the factory object provided by the server and passes the returned capability of that request as `"foo_service"` to the client process.

Note

The `svc:create(..)` call blocks on the server. This means the script execution blocks until the my-service application handles the create request.

4.3 Memory management - Data Spaces and the Region Map

4.3.1 User-level paging

Memory management in L4-based systems is done by user-level applications, the role is usually called *pager*. Tasks can give other tasks full or restricted access rights to parts of their own memory. The kernel offers means to grant the memory in a secure way, often referred to as *memory mapping*.

The described mechanism can be used to construct a memory hierarchy among tasks. The root of the hierarchy is `sigma0`, which initially gets all system resources and hands them out once on a first-come-first-served basis. Memory resources can be mapped between tasks at a page-size granularity. This size is predetermined by the CPU's memory management unit and is commonly set to 4 kB.

4.3.1.1 Data spaces

A data space is the [L4Re](#) abstraction for objects which may be accessed in a memory mapped fashion (i.e., using normal memory read and write instructions). Examples include the sections of a binary which the loader attaches to the application's address space, files in the ROM or on disk provided by a file server, the registers of memory-mapped devices and anonymous memory such as the heap or the stack.

Anonymous memory data spaces in particular (but in general all data spaces except memory mapped IO) can either be constructed entirely from a portion of the RAM or the current working set may be multiplexed on some portion of the RAM. In the first case it is possible to eagerly insert all pages (more precisely page-frame capabilities) into the application's address space such that no further page faults occur when this data space is accessed. In general, however, only the pages for the some portion are provided and further pages are inserted by the pager as a result of page faults.

4.3.1.2 Virtual Memory Handling

The virtual memory of each task is constructed from data spaces, backing virtual memory regions (VMRs). The management of the VMRs is provided by an object called *region map*. A dedicated region-map object is associated with each task, it allows to attach and detach data spaces to an address space as well as to reserve areas of virtual memory. Since the region-map object possesses all knowledge about virtual memory layout of a task, it also serves as an application's default pager.

4.3.1.3 Memory Allocation

Operating systems commonly use anonymous memory for implementing dynamic memory allocation (e.g., using `malloc` or `new`). In an L4Re-based system, each task gets assigned a memory allocator providing anonymous memory using data spaces.

See also

[L4Re::Dataspace](#) and [L4Re::Rm](#).

4.4 Program Input and Output

The initial environment provides a Vcon capability used as the standard input/output stream.

Output is usually connected to the parent of the program and displayed as debugging output. The standard output is also used as a back end to the C-style `printf` functions and the C++ streams.

Vcon services are implemented in Moe and the loader as well as by the Fiasco kernel and connected either to the serial line or to the screen if available.

See also

[Virtual Console](#)

4.5 Initial Memory Allocator and Factory

The purpose of the memory allocator and of the factory is to provide the application with the means to allocate memory (in the form of data spaces) and kernel objects respectively.

An initial memory allocator and an initial factory are accessible via the allocation [L4Re](#) environment.

See also

[L4Re::Mem_alloc](#)

The factory is a kernel object that provides the ability to create new kernel objects dynamically. A factory imposes a resource limit for kernel memory, and is thus a means to prevent denial of service attacks on kernel resources. A factory can also be used to create new factory objects.

See also

[Factory](#)

4.6 Application and Server Building Blocks

So far we have discussed the environment of applications in which a single thread runs and which may invoke services provided through their initial objects.

In the following we describe some building blocks to extend the application in various dimensions and to eventually implement a server which implements user-level objects that may in turn be accessed by other applications and servers.

4.6.1 Creating Additional Application Threads

To create application threads, one must allocate a stack on which this thread may execute, create a thread kernel object and setup the information required at startup time (instruction pointer, stack pointer, etc.). In L4Re this functionality is encapsulated in the pthread library.

4.6.2 Providing a Service

In capability systems, services are typically provided by transferring a capability to those applications that are authorised to access the object to which the capability refers to.

Let us discuss an example to illustrate how two parties can communicate with each other: Assume a simple file server, which implements an interface for accessing individual files: `read(pos, buf, length)` and `write(pos, data, length)`.

L4Re provides support for building servers based on the class `L4::Server_object`. `L4::Server_object` provides an abstract interface to be used with the `L4::Server` class. Specific server objects such as, in our case, files inherit from `L4::Server_object`. Let us call this class `File_object`. When invoked upon receiving a message, the `L4::Server` will automatically identify the corresponding server object based on the capability that has been provided to its clients and invoke this object's `dispatch` function with the incoming message as a parameter. Based on this message, the server must then decide which of the protocols it implements was invoked (if any). Usually, it will evaluate a protocol specific opcode that clients are required to transmit as one of the first words in the message. For example, assume our server assigns the following opcodes: `Read = 0` and `Write = 1`. The `dispatch` function calls the corresponding server function (i.e., `File_object::read()` or `File_object::write()`), which will in turn parse additional parameters given to the function. In our case, this would be the position and the amount of data to be read or written. In case the write function was called the server will now update the contents of the file with the data supplied. In case of a read it will store the requested part of the file in the message buffer. A reply to the client finishes the client request.

4.7 Pthread Support

L4Re supports the standard pthread library functionality.

Therefore L4Re itself does not contain any documentation for pthreads itself. Please refer to the standard pthread documentation instead.

The L4Re specific parts will be described herein.

- Include pthread-l4.h header file:

```
#include <pthread-l4.h>
```

- Return the local thread capability of a pthread thread:

Use `pthread_l4_cap(pthread_t t)` to get the capability index of the pthread t.

For example:

```
pthread_l4_cap(pthread_self());
```

- Setting the L4 priority of an L4 thread works with a special scheduling policy (other policies do not affect the L4 thread priority):

```
pthread_t t;
pthread_attr_t a;
struct sched_param sp;

pthread_attr_init(&a);
sp.sched_priority = l4_priority;
pthread_attr_setschedpolicy(&a, SCHED_L4);
pthread_attr_setschedparam(&a, &sp);
pthread_attr_setinheritsched(&a, PTHREAD_EXPLICIT_SCHED);

if (pthread_create(&t, &a, pthread_func, NULL))
    // failure...

pthread_attr_destroy(&a);
```

- You can prevent your pthread from running immediately after the call to `pthread_create(..)` by adding `PTHREAD_L4_ATTR_NO_START` to the `create_flags` of the pthread attributes. To finally start the thread you need to call `scheduler()->run_thread()` passing the capability of the pthread and scheduling parameters.

```
pthread_t t;
pthread_attr_t attr;

pthread_attr_init(&attr);
attr.create_flags |= PTHREAD_L4_ATTR_NO_START;

if (pthread_create(t, &attr, pthread_func, nullptr))
    // failure...

pthread_attr_destroy(&attr);

// do stuff

auto ret = L4Re::Env::env()->scheduler()->run_thread(pthread_l4_cap(t),
                                                    l4_sched_param(2));
if (l4_error(ret))
    // failure...
```

4.8 Interface Definition Language

An interface definition in [L4Re](#) is normally declared as a class derived from [L4::Kobject_t](#).

For example, the simple calculator example declares its interface like that:

```
struct Calc : L4::Kobject_t<Calc, L4::Kobject>
{
    L4_INLINE_RPC(int, sub, (l4_uint32_t a, l4_uint32_t b,
                          l4_uint32_t *res));
    L4_INLINE_RPC(int, neg, (l4_uint32_t a, l4_uint32_t *res));

    typedef L4::Typeid::Rpc<sub_t, neg_t> Rpc;
};
```

The signature of each function is first declared using one of the RPC macros (see below) and then all the functions need to be listed in the `Rpcs` type.

4.8.1 Parameter types for RPC

Generally all value parameters, const reference parameters, and const pointer parameters to an RPC interface are considered as input parameters for the RPC and are transmitted from the client to the server.

Note

This means that `char const *` is treated as an input `char` and not as a zero terminated string value, for strings see `L4::lpc::String<>`.

Parameters that are non-const references or non-const pointers are treated as output parameters going from the server to the client.

```
L4_RPC(long, test, (int arg1, char const *arg2, unsigned *ret1));
```

The example shows the declaration of a method called `test` with `long` as return type, `arg1` is an `int` input, `arg2` a `char` input, and `ret1` an unsigned output parameter.

Type	Direction	Client-Type	Server-Type
T	Input	T	T
T const &	Input	T const &	T const &
T const *	Input	T const *	T const &
T &	Output	T &	T &
T *	Output	T *	T &
L4::Ipc::In_out<T &>	In/Out	T &	T &
L4::Ipc::In_out<T *>	In/Out	T *	T &
L4::Ipc::Cap<T>	Input	L4::Ipc::Cap<T>	L4::Ipc::Snd_fpage
L4::Ipc::Out<L4::Cap<T> >	Output	L4::Cap<T>	L4::Ipc::Cap<T> &

Array types can be used to transmit arrays of variable length. They can either be stored in a client-provided buffer (L4::Ipc::Array), copied into a server-provided buffer (L4::Ipc::Array_in_buf) or directly read and written into the U↔TCB (L4::Ipc::Array_ref). For latter type, the start position of an array type needs to be known in advance. That implies that only one such array can be transmitted per direction and it must be the last part in the message.

Type	Direction	Client-Type	Server-Type
L4::Ipc::↔ Array<const T>	Input	L4::Ipc::↔ Array<const T>	L4::Ipc::Array_↔ ref<const T>
L4::Ipc::↔ Array<const T>	Input	L4::Ipc::↔ Array<const T>	L4::Ipc::Array_in_↔ buf<T> const &
L4::Ipc::Array<T> &	Output	L4::Ipc::Array<T> &	L4::Ipc::Array_↔ ref<T> &
L4::Ipc::Array_↔ ref<T> &	Output	L4::Ipc::Array_↔ ref<T> &	L4::Ipc::Array_↔ ref<T> &

Finally, there are some optional types where the sender can choose if the parameter should be included in the message. These types are for the implementation of some legacy message formats and should generally not be needed for the definition of ordinary interfaces.

Type	Direction	Client-Type	Server-Type
L4::Ipc::Opt<T>	Input	L4::Ipc::Opt<T>	T
L4::Ipc::Opt<const T*>	Input	L4::Ipc::Opt<const T*>	T
L4::Ipc::Opt<T &>	Output	T &	L4::Ipc::Opt<T> &
L4::Ipc::Opt<T *>	Output	T *	L4::Ipc::Opt<T> &
L4::Ipc::Opt<Array_↔ _ref<T> &>	Output	Array_ref<T> &	L4::Ipc::Opt<Array_↔ _ref<T>> &

4.8.2 RPC Return Types

On the server side, the return type of an RPC handling function is always `long`. The return value is transmitted via the label field in `l4_msgtag_t` and is therefore restricted to its length. Per convention, a negative return value is interpreted as an error condition. If the return value is negative, output parameters are not transmitted back to the client.

Attention

The client must never rely on the content of output parameters when the return value is negative.

On the client-side, the return value of the RPC is set as defined in the RPC macro. If `l4_msgtag_t` is given, then the client has access to the full message tag, otherwise the return type should be `long`. Note that the client might not only receive the server return value in response but also an IPC error code.

4.8.3 RPC Method Declaration

RPC member functions can be declared using one of the following C++ macros.

For inline RPC stubs, where the RPC stub code itself is `inline`:

- `L4_INLINE_RPC(res, name, (args...), flags)`
Define an inline RPC call (type and callable).
- `L4_INLINE_RPC_OP(op, res, name, (args...), flags)`
Define an inline RPC call with specific opcode (type and callable).
- `L4_INLINE_RPC_NF(res, name, (args...), flags)`
Define an inline RPC call type (the type only, no callable).
- `L4_INLINE_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`
Define an inline RPC call type with specific opcode (the type only, no callable).

For external RPC stubs, where the RPC stub code must be defined in a separate compile unit (usually a `.cc` file):

- `L4_RPC(Ret_type, func_name, (args...), flags)`
Define an RPC call (type and callable).
- `L4_RPC_OP(opcode, Ret_type, func_name, (args...), flags)`
Define an RPC call with specific opcode (type and callable).
- `L4_RPC_NF(Ret_type, func_name, (args...), flags)`
Define an RPC call type (the type only, no callable).
- `L4_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`
Define an RPC call type with specific opcode (the type only, no callable).

To generate the implementation of an external RPC stub:

- `L4_RPC_DEF(class_name::func_name)`
Generate the definition of an RPC stub.

The `NF` versions of the macro generally do not generate a callable member function named `<name>` but do only generate the type `<name>_t`. This data type can be used to call the RPC stub explicitly using `<name>_t::call(L4::Cap<Iface_class> cap, args...)`.

4.9 L4Re Build System

L4Re uses a custom make-based build system, often simply referred to as *BID*.

This section explains how to use BID when writing applications and libraries for L4Re.

4.9.1 Building L4Re

Setting up the Build Directory

L4Re must be built out-of-source. Therefore the first mandatory step is creating and populating a build directory. From the root of the L4Re source tree run

```
make B=<builddir>
```

Other targets that can be executed in the source directory are

update

Update the source directory from svn. Only makes sense when you have downloaded L4Re from the official subversion repository.

help

Show a short help with the most important targets.

Invoking Make

Once the build directory is set up, BID make can be invoked in one of two ways:

1. Go to the build directory and invoke make without special options.
2. Go to a source directory with a BID make file and invoke `make O=<builddir> . . .`

The default target builds the source (as you would expect), other targets that are available in build mode are

cleanfast

Quickly cleans the build directory by removing all subdirectories that contain generated files. The configuration will remain untouched.

clean

Remove generated files. Slower than `make cleanfast` but can be used on selected packages. Use `S= . . .` to select the target package.

In addition to these targets, there are a number of targets to generate images which are explained elsewhere.

4.9.2 Writing BID Make Files

The BID build system exports different roles that define what should be done in the subdirectory. So a BID make file essentially consists of defining the role and a number of role-dependent make variables. The basic layout should look like this:

```
PKGDIR  ?= <path to package's root directory> # e.g., '.' or '..'
L4DIR   ?= <path to L4Re source directory>    # e.g. '$(PKGDIR)/../../'

<various definitions>

include $(L4DIR)/mk/<role>.mk
```

PKGDIR in the first line defines the root directory of the current package. L4DIR in the next line must be pointed to the root of the [L4Re](#) source tree the package should be built against. After this custom variable definitions for the role follow. In the final line of the file, the make file with the role-specific rules must be sourced.

The following roles are currently defined:

- project.mk - Sub-project Role
- subdir.mk - Directory Role
- [prog.mk](#) - Application Role
- lib.mk - Library Role
- [include.mk](#) - Header File Role
- doc.mk - Documentation Role
- [test.mk](#) - Test Application Role
- idl.mk - IDL File Role (currently unused)
- runux.mk - Tests in FiascoUX Role

BID-global Variables

This section lists variables that configure how the BID build system behaves. They are applicable for all roles.

Variable	Description
CC	C compiler for target
CXX	C++ compiler for target
HOST_CC	C compiler for host
HOST_CXX	C++ compiler for host

4.9.3 prog.mk - Application Role

The prog role is used to build executable programs.

General Configuration Variables

The following variables can only be set globally for the Makefile:

MODE

Kind of target to build for. The following values are possible:

- `static` - build a statically linked binary (default)
- `shared` - build a dynamically linked binary
- `l4linux` - build a binary for running on L4Linux on the target platform
- `host` - build for host system
- `targetsys` - build a binary for the target platform with the compiler's default settings

SYSTEMS

List of architectures the target can be built for. The entries must be space-separated entries either naming an architecture (e.g. `amd64`) or an architecture and ABI (e.g. `arm-l4f`). When not defined, the target will be built for all possible platforms.

TARGET

Name or names of the binaries to compile. This variable may also be postfixed with a specific architecture.

SRC_CC_IS_CXX11

C++ standard to use. Default is `c++0x`.

Target-specific Configuration Variables

The following variables may either be used with or without a description suffix. Without suffix they will be used for all operations. With a specific description their use is restricted to a subset. These specifications include a target file and the architecture, both optional but in this order, separated by underscores. The specific variables will be used in addition to the more general ones.

SRC_C / SRC_CC / SRC_F / SRC_S

`.c`, `.cc`, `.f90`, `.S` source files.

REQUIRES_LIBS

List of libraries the binary depends on. This works only with libraries that export a pkg_config configuration file. Automatically adds any required include and link options.

DEPENDS_PKGS

List of packages this binary depends on. If one these packages is missing then building of the binary will be skipped.

CPPFLAGS / CFLAGS / CXXFLAGS / FFLAGS / ASFLAGS

Options for the C preprocessor, C compiler, C++ compiler, Fortran compiler and assembler. When used with suffix, the referred element is the source file, not the target file.

LDFLAGS

Options for the linker ld.

LIBS

Additional libraries to link against (with -l).

PRIVATE_LIBDIR

Additional directories to search for libraries.

CRT0 / CRTN

(expert use only) Files containing custom startup and finish code.

LDSCRIPT

(expert use only) Custom link script to use.

4.9.4 include.mk - Header File Role

The header file role is responsible for installing header file at the appropriate location.

The following variables can be used for customizing the process:

INCSRC_DIR

Source directory where the headers can be found. Default is the directory where the Makefile resides.

TARGET

List of header files to install. If left undefined, then `INCSRC_DIR` will be scanned for files with suffix `.h` or `.i`.

EXTRA_TARGET

When `TARGET` is undefined, then add these files to the headers found by scanning the source directory. Has no effect if `TARGET` has been defined.

CONTRIB_HEADERS

When set, the headers will be installed in `${BUILDDIR}/include/contrib/${PKGNAME}` rather than `${BUILDDIR}/include/l4/${PKGNAME}`.

INSTALL_INC_PREFIX

Base directory where to install the headers. Overwrites `CONTRIB_HEADERS`. The headers will then be found under `${BUILDDIR}/include/${INSTALL_INC_PREFIX}`.

PC_FILENAME

When set, a `pkg_config` configuration file is created with the given name.

4.9.5 test.mk - Test Application Role

The test role is very similar to the application role, it also builds an executable binary.

The difference is that it also builds for each target a test script that executes the test target either on the host (MODE=host) or a target platform (currently only qemu).

The role accepts all make variables that are accepted by the application role. The only difference is that the `TARGET` variable is not required. If it is missing, the source directory will be scanned for source files that fit the pattern `test_*.c[c]` and create one target for each of them.

Note

It is possible to still use `SRC_C[C]` when targets are determined automatically. In that case the specified sources will be used in addition* to the main `test_*.c[c]` source.

In addition to the variables above, there are a number of variables that control how the test is executed. All these variables may be used as a global variable that applies to all test or, if the target name is added as a suffix, set for a specific target only.

TEST_TARGET

Name of binary containing the test (default: same as `TARGET`).

TARGET_\$(ARCH)

When `TARGET` is undefined, these targets are added to the list of targets for the specified architecture. For all targets `SRC_C[C]` files must be defined separately.

TEST_EXPECTED

File containing expected output. By default the variable is empty, which means the test binary is expected to produce TAP test output, that can be directly processed.

TEST_EXPECTED_REPEAT

Number of times the expected output should be repeated, by default 1. When set to 0 then output is expected to repeat forever. This is particularly useful to make sure that stress tests that are meant to run in an endless loop are still alive. Note that such endless tests can only be run by directly executing the test script. They will be skipped when run in a test harness like `prove`.

TEST_TIMEOUT

Non-standard timeout after which the test run is aborted (useful for tests involving sleep).

NED_CFG

LUA configuration file for startup to give to Ned

REQUIRED_MODULES

Additional modules needed to run the test.

QEMU_ARGS

Additional parameters to supply to QEMU.

MOE_ARGS

Additional parameters to supply to moe.

KERNEL_CONF

Features the Fiasco kernel must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` looks for a `globalconfig.out` file in the same directory as the kernel and checks that all options are enabled. If not, the test is skipped. Has only an effect if the `globalconfig.out` file is present.

L4LINUX_CONF

Features the L4Linux kernel must have been compiled with. Similar to `KERNEL_CONF` but checks for a `.config` file in the directory of the L4Linux kernel.

TEST_SETUP

Command to execute before the test is run. The test will only be executed if the command returns 0. If the exit code is 2, the test is marked as skipped with the reason provided in the final line of stdout.

TEST_LOGFILE

Append output of test execution to the given file unless `TEST_WORKDIR` is given.

TEST_WORKDIR

Create logs, temp and other files below the given directory. That directory is taken as base dir for more automatically created subdir levels using the current test path, in order to guarantee conflict-free usage when running many different tests with a common workdir. When `TEST_WORKDIR` is provided then `TEST_LOGFILE` is ignored as it is organized below workdir.

TEST_TAGS

Comma separated list of tags which have to be specified in order to run the test. Test will be skipped without error otherwise. Useful to mark tests, e.g. broken or slow. Tag names can basically contain any character except comma. Whitespace characters surrounding a tag name will be ignored. A tag starting with exclamation mark is checked to not appear in the specified run tags, otherwise the test is also skipped.

In addition to compiled tests, it is also possible to create tests where the test binary or script comes from a different source. These tests must be listed in `EXTRA_TARGET` and for each target a custom `TEST_TARGET` must be provided.

Running Tests

The make role creates a test script which can be found in `<builddir>/test/t/<arch>/<api>`. It is possible to organise the tests further in subdirectories below by specifying a `TEST_GROUP`.

To be able to execute the test, a minimal test environment needs to be set up by exporting the following environment variables:

KERNEL_<arch>, KERNEL

Fiasco kernel binary to use. The test runner is able to check if the kernel has all features necessary for the test and skip tests accordingly. In order for this to work, the `globalconfig.out` config file from the build directory needs to be available in the same directory as the kernel.

L4LX_KERNEL_<arch>, L4_LX_KERNEL

L4Linux binary to use. This is only required to run tests in `mode=l4linux`. If no L4Linux kernel is set then these tests will simply be skipped. The test runner is also able to check if the kernel has all features compiled in that are required to run the test successfully (see make variable `L4LINUX_CONF` above). For this to work, the `.config` configuration file from the build directory needs to be available in the same directory as the kernel.

LINUX_RAMDISK_<arch>, LINUX_RAMDISK

Ramdisk to mount as root in L4Linux. This is only required to run tests in `mode=l4linux`. If not supplied, L4Linux tests will be skipped. The ramdisk must be set up to start the test directly after the initial startup is finished. The name of the test binary is supplied via the kernel command line option `l4re_testprog`. The `tool/test` directory contains an example script `launch-l4linux-test`, which can be copied onto the ramdisk and started by the init script.

In addition to these variables, the following BID variables can be overwritten at runtime: `PT` (for the platform type) and `TEST_TIMEOUT`. You may also supply `QEMU_ARGS` and `MOE_ARGS` which will be appended to the parameters specified in the BID test make file.

Once the environment is set up, the tests can be run either by simply executing all of them from the build directory with

```
make test
```

or executing them directly, like

```
test/t/amd64_amdfam10/l4f/l4re-core/moe/test_namespace.t
```

or running one or more tests through the test harness `prove`, like

```
prove test/t/amd64_amdfam10/l4f/l4re-core/moe/test_namespace.t
prove -r test/t/amd64_amdfam10/l4f/l4re-core/
prove -rv test/t/amd64_amdfam10/l4f/l4re-core/
```

To run tests with tags (see `TEST_TAGS`), these tags need to be specified either in the environment variable `TEST_RUN_TAGS` or as arguments to the `test.t` files:

```
$ test/t/amd64_amdfam10/l4f/l4re-core/test_one.t --run-tags slow,gtest-shuffle
$ test/t/amd64_amdfam10/l4f/l4re-core/test_one.t -T slow,gtest-shuffle
$ prove -r test/t/amd64_amdfam10/l4f/l4re-core/ : -T slow,gtest-shuffle
$ TEST_RUN_TAGS=slow,gtest-shuffle prove -r test/t/amd64_amdfam10/l4f/l4re-core/
```

Without providing run tags, tests requiring certain tags are skipped:

```
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... skipped: requires additional run tags: slow
test/t/amd64_amdfam10/test_three.t .. ok
```

When tags are provided, the tests requiring those tags are now also executed, the tests that forbid them are skipped:

```
$ TEST_RUN_TAGS=slow,gtest-shuffle
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... ok
test/t/amd64_amdfam10/test_three.t .. skipped: not supported with run tags: gtest-shuffle
```

Tests without any tags are always executed.

Running Tests in External Programs

You can hand-over test execution to an external program by setting the environment variable `EXTERNAL_TEST_STARTER` to the full path of that program:

```
export EXTERNAL_TEST_STARTER=/path/to/external/test-starter
make test
```

EXTERNAL_TEST_STARTER

This variable is evaluated by `tool/bin/run_test` (the backend behind `make test`) and contains the full path to the tool which actually starts the test instead of the test itself.

The `EXTERNAL_TEST_STARTER` can be any program instead of the default execution via `make qemu E=maketest`. Its output is taken by `run_test` as the test output.

Usually it is just a bridge to prepare the test execution, e.g., it could create the test as image and start that image via a simulator.

Running Tests in a Simulator

Based on above mechanism there is a dedicated external test starter `tool/bin/teststarter-image-telnet.pl` shipped in BID which assumes an image to be started with another program which provides test execution output on a network port.

This can be used to execute tests in a simulator, like this:

```
export EXTERNAL_TEST_STARTER=$L4RE_SRC/tool/bin/teststarter-image-telnet.pl
export SIMULATOR_START=/path/to/configured/simulator-exe
make test
```

After building the image and starting the simulator it contacts the simulator via a network port (sometimes called "telnet" port) to pass-through its execution output as its own output so it gets captured by `run_test` as usual.

The following variables control `teststarter-image-telnet.pl`:

SIMULATOR_START

This points to the full path of the program that actually starts the prepared test image. Most often this is the frontend script of your simulator environment which is pre-configured so that it actually works in the way that `teststarter-image-telnet.pl` expects from the following settings.

SIMULATOR_IMAGETYPE

The image type to be generated via `make $SIMULATOR_IMAGETYPE E=maketest`. Default is `elfimage`.

SIMULATOR_HOST

The simulator will be contacted via socket on that host to read its output. Default is `localhost`.

SIMULATOR_PORT

The simulator will be contacted via socket on that port to read its output. Default is `11111`.

SIMULATOR_START_SLEEPTIME

After starting the simulator it waits that many seconds before reading from the port. Default is `1` (second).

Running tests without tapper-wrapper

In case you want to replace the tapper-wrapper test starter, you can replace the default one by setting the environment variable `TEST_STARTER` to the path of your test starter. Then your test starter can use the same environment which is normally set up for the default starter, which includes environment variables provided by the build system as well as the test itself. Among these are `SEARCHPATH`, `MODE`, `ARCH`, `MOE_CFG`, `MOE_ARGS`, `TEST_TIMEOUT`, `TEST_TARGET`, `TEST_EXPECTED`, `QEMU_ARGS` and many more.

Debugging Tests

The test script is only a thin wrapper that sets up the test environment as it was defined in the make file and then executes two scripts: `tapper-wrapper` and `run_test`.

The main work horse of the two is `tool/bin/run_test`. It collects the necessary files and starts qemu to execute the test. This script is always required.

There is then a second script wrapped around the test runner: `tool/bin/tapper-wrapper`. This tool inspects the output of the test runner and reformats it, so that it can be read by tools like `prove`. If the test produces tap output, then the script scans for this output and filters away all the debug output. If `TEST_EXPECTED` was defined, then the script scans the output for the expected lines and prints a suitable TAP message with success or failure. It also makes sure that qemu is killed as soon as the test is finished.

There are a number of command-line parameters that allow to quickly change test parameters for debugging purposes. Run the test with `'-help'` for more information about available parameters.

Chapter 5

L4Re Servers

Here you shall find a quick overview over the standard services running on Fiasco.OC and [L4Re](#).

Sigma0, the Root Pager

Sigma0 is a special server running on [L4](#) because it is responsible of resolving page faults for the root task, the first useful task on [L4Re](#). Sigma0 can be seen as part of the kernel, however it runs in unprivileged mode. To run something useful on Fiasco.OC you usually need to run Sigma0, nevertheless it is possible to replace Sigma0 by a different implementation.

Moe, the Root Task

Moe is our implementation of the [L4](#) root task that is responsible for bootstrapping the system, and to provide basic resource management services to the applications on top. Therefore Moe provides [L4Re](#) resource management and multiplexing services:

- **Memory** in the form of memory allocators ([L4Re::Mem_alloc](#), [L4::Factory](#)) and data spaces ([L4Re::↔Dataspace](#))
- **Cpu** in the form of basic scheduler objects ([L4::Scheduler](#))
- **Vcon** multiplexing for debug output (output only)
- **Virtual memory management** for applications, [L4Re::Rm](#)

Moe further provides an implementation of [L4Re](#) name spaces ([L4Re::Namespace](#)), which are for example used to provide a read only directory of all multi-boot modules. In the case of a boot loader, like grub that enables a VESA frame buffer, there is also a single instance of an [L4Re](#) graphics session ([L4Re::Goos](#)).

To start the system Moe starts a single ELF program, the init process. The init process (usually Ned, see the next section) gets access to all resources managed by Moe and to the Sigma0 root pager interface.

For more details see [Moe, the Root-Task](#).

Ned, the Default Init Process

To keep the root task free from complicated scripting engines and to avoid circular dependencies in application startup (that could lead to dead locks) the configuration and startup of the real system is managed by an extra task, the init process.

Ned is such an init process that allows system configuration via Lua scripts.

For more information see [Ned](#).

Io, the Platform and Device Resource Manager

Because all peripheral management of Fiasco.OC is done in user-level applications, there is the need to have a centralized management of the resources belonging to the platform and to peripheral devices.

This is the job of Io. Io provides portable abstractions for iterating and accessing devices and their resources (IRQ's, IO Memory...), as well as delegating access to those resources to other applications (e.g., device drivers).

For more details see [Io, the Io Server](#).

Other Servers

The following additional server package are available on top of the core [L4Re](#) environment.

- **Rtc, the Real-Time Clock Server**
is a simple multiplexer for real-time clock hardware on your platform.
- **fb-drv, the Low-Level Graphics Driver**
provides low-level access and initialization of various graphics hardware. It has support for running VESA BIOS calls on Intel x86 platforms, as well as support for various ARM display controllers. `fb-drv` provides a single instance of the L4Re::Goos interface and can serve as a back end for the Mag server, in particular, if there is no graphics support in the boot loader.
- [Uvmm, the virtual machine monitor](#)
- [Mag, the GUI Multiplexer](#)
Our default multiplexer for the graphics hardware is Mag. Mag is a Nitpicker (TODO: ref) derivate that allows secure multiplexing of the graphics and input hardware among multiple applications and multiple complete windowing environments.
- [Cons, the Console Multiplexer](#)
An interactive multiplexer for console in- and output. It buffers the output from different L4 clients and allows to switch between them to redirect input.

5.1 Moe, the Root-Task

Moe is the default root-task implementation for L4Re-based systems.

Moe is the first task which is usually started in L4Re-based systems. The micro kernel starts *Moe* as the Root-Task.

Moe provides a default implementation for the basic L4Re abstractions, such as data spaces (L4Re::Dataspace), region maps (L4Re::Rm), memory allocators (L4Re::Mem_alloc, L4::Factory), name spaces (L4Re::Namespace) and so on (see L4Re Interface).

Moe consists of the following subsystems:

- [Name-Space Provider](#) — provides instances of name spaces
- [Boot FS](#) — provides access to the files loaded during platform boot (e.g., linked into the boot image or loaded via GRUB boot loader)
- [Log Subsystem](#) — provides tagged log output for applications
- [Scheduler Subsystem](#) — provides simple scheduler objects for scheduling policy enforcement
- [Memory Allocator, Generic Factory](#) — provides allocation of physical RAM as data spaces, as well as allocation of the other L4Re objects provided by Moe

5.1.1 Memory Allocator, Generic Factory

The generic factory in Moe is responsible for all kinds of dynamic object allocation. The interface is a combination of L4::Factory and, for traditional reasons, L4Re::Mem_alloc.

The generic factory interface allows allocation of the following objects:

- [L4Re::Namespace](#)
- [L4Re::Dataspace](#), RAM allocation
- [L4Re::Dma_space](#), memory management for DMA-capable devices
- [L4Re::Rm](#), Virtual memory management for application tasks
- [L4::Vcon](#) (output only)
- [L4::Scheduler](#), to provide a restricted priority / CPU range for clients
- [L4::Factory](#), to provide a quota limited allocation for clients

The memory allocator in Moe is the alternative interface for allocating memory (RAM) in terms of L4Re::Dataspace-s (see also L4Re::Mem_alloc).

Dataspaces can be allocated with an arbitrary size. The granularity for memory allocation however is the machine page size (L4_PAGESIZE). A dataspace user must be aware that, as a result of this page-size granularity, there may be padding memory at the end of a dataspace which is accessible to each client. Moe currently allows most dataspace operations on this padding area. Nonetheless, the client must not make any assumptions about the size or content of the padding area, as this behaviour might change in the future.

The provided data spaces can have different characteristics:

- Physically contiguous and pre-allocated
- Non contiguous and on-demand allocated with possible copy on write (COW)

5.1.2 Name-Space Provider

Moe provides a name spaces conforming to the [L4Re::Namespace](#) interface (see [Name-space API](#)). Per default Moe creates a single name space for the [Boot FS](#). That is available as `rom` in the initial objects of the init process.

5.1.2.1 Boot FS

The Boot FS subsystem provides access to the files loaded during the platform boot (or available in ROM). These files are either linked into the boot image or loaded via a flexible boot loader, such as GRUB.

The subsystem provides an [L4Re::Namespace](#) object as directory and an [L4Re::Dataspace](#) object for each file.

By default all files are read only and visible in the namespace `rom`. As an option, files can be supplied with the argument `:rw` to mark them as writable modules. Moe will allow read and write access to these dataspace and make them visible in a different namespace called `rwfs`.

An example entry in 'modules.list' would look like this:

```
module somemodule :rw
```

Note

In order for a client to receive write permissions to the dataspace, the corresponding cap also needs write permissions.

5.1.3 Log Subsystem

The logging facility of Moe provides per application tagged and synchronized log output.

5.1.4 Scheduler Subsystem

The scheduler subsystem provides a simple scheduler proxy.

5.1.5 Command-Line Options

Moe's command-line syntax is:

```
moe [--debug=<flags>] [--init=<binary>] [--l4re-dbg=<flags>] [--ldr-flags=<flags>] [-- <init options>]
```

`--debug=<debug flags>`

This option enables debug messages from Moe itself, the `<debug flags>` values are a combination of `info`, `warn`, `boot`, `server`, `loader`, `exceptions`, and `ns` (or `all` for full verbosity).

`--init=<init process>`

This options allows to override the default init process binary, which is 'rom/ned'.

`--l4re-dbg=<debug flags>`

This option allows to set the debug options for the [L4Re](#) runtime environment of the init process. The flags are the same as for `--debug=`.

`--ldr-flags=<loader flags>`

This option allows setting some loader options for the [L4Re](#) runtime environment. The flags are `pre_alloc`, `all_segs_cow`, and `pinned_segs`.

`-- <init options>`

All command-line parameters after the special `--` option are passed directly to the init process.

5.2 Ned, the Init Process

Ned's job is to bootstrap the system running on [L4Re](#).

The main thing to do here is to coordinate the startup of services and applications as well as to provide the communication channels for them. The central facility in Ned is the Lua (<http://www.lua.org>) script interpreter with the [L4Re](#) and ELF-loader bindings.

The boot process is based on the execution of one or more Lua scripts that create communication channels (IPC gates), instantiate other [L4Re](#) objects, organize capabilities to these objects in sets, and start application processes with access to those objects (or based on those objects).

For starting applications, Ned depends on the services of [Moe, the Root-Task](#) or another *loader*, which must provide data spaces and region maps. Ned also uses the 'rom' capability as source for Lua scripts and at least the 'l4re' binary (the runtime environment core) running in each application.

Each application Ned starts is equipped with an [L4Re::Env](#) environment that provides information about all the initial objects made accessible to this application.

5.2.1 Lua Bindings for L4Re

Ned provides various bindings for [L4Re](#) abstractions. These bindings are located in the '[L4](#)' package (`require "L4"`).

5.2.1.1 Capabilities in Lua

Capabilities are handled as normal values in Lua. They can be stored in normal variables or Lua compound structures (tables). A capability in Lua possesses additional information about the access rights that shall be transferred to other tasks when the capability is transferred. To support implementation of the Principle of Least Privilege, minimal rights are assigned by default. Extended rights can be added using the method `mode("...")` (short `m("...")`) that returns a new reference to the capability with the given rights.

Note

It is generally impossible to elevate the real access rights to an object. This means that if Ned has only restricted rights to an object it is not possible to upgrade the access rights with the `mode` method.

The capabilities in Lua also carry dynamic type information about the referenced objects. They thereby provide type-specific operations on the objects, such as the `create` operation on a generic factory or the `query` and `register` operations on a name space.

5.2.1.2 Access to L4Re::Env Capabilities

The initial objects provided to Ned itself are accessible via the table `L4.Env`. The default (usually unnamed) capabilities are accessible as `factory`, `log`, `mem_alloc`, `parent`, `rm`, and `scheduler` in the `L4.Env` table.

5.2.1.3 Constants

Protocols

The protocol constants are defined by default in the [L4](#) package's table `L4.Proto`. The definition is not complete and only covers what is usually needed to configure and start applications. The protocols are for example used as first argument to the `Factory:create` method.

```
Proto = {
  Dataspace = 0x4000,
  Namespace = 0x4001,
  Goos       = 0x4003,
  Mem_alloc = 0x4004,
  Rm         = 0x4005,
  Event      = 0x4006,
  Inhibitor  = 0x4007,
  Sigma0     = -6,
  Log        = -13,
  Scheduler  = -14,
  Factory    = -15,
  Vm         = -16,
  Dma_space  = -17,
  Irq_sender = -18,
  Irq_muxer  = -19,
  Semaphore  = -20,
  Iommu      = -22,
  Ipc_gate   = 0,
}
```

Debugging Flags

Debugging flags used for the applications [L4Re](#) core:

```
Dbg = {
  Info      = 1,
  Warn      = 2,
  Boot      = 4,
  Server    = 0x10,
  Exceptions = 0x20,
  Cmd_line  = 0x40,
  Loader    = 0x80,
  Name_space = 0x400,
  All       = 0xffffffff,
}
```

Loader Flags

Flags for configuring the loading process of an application.

```
Ldr_flags = {
  eager_map    = 0x1, -- L4RE_AUX_LDR_FLAG_EAGER_MAP
  all_segs_cow = 0x2, -- L4RE_AUX_LDR_FLAG_ALL_SEGS_COW
  pinned_segs  = 0x4, -- L4RE_AUX_LDR_FLAG_PINNED_SEGS
}
```

5.2.1.4 Application Startup Details

The central facility for starting a new task with Ned is the class `L4.Loader`. This class provides interfaces for conveniently configuring and starting programs. It provides three operations:

- `new_channel()` Returns a new IPC gate that can be used to connect two applications
- `start()` and `startv()` Start a new application process and return a process object

The `new_channel()` call is used to provide a service application with a communication channel to bind its initial service to. The concrete behavior of the object and the number of IPC gates required by a server depends on the server implementation. The channel can be passed to client applications as well or can be used for operations within the script itself.

`start()` and `startv()` always require at least two arguments. The first one is a table that contains information about the initial objects an application shall get. The second argument is a string, which for `start()` is the program name plus a white-space-separated list of program arguments (`argv`). For `startv()` the second argument is just the program binary name – which may contain spaces –, and the program arguments are provided as separate string arguments following the binary name (allowing spaces in arguments, too). The last optional argument is a table containing the POSIX environment variables for the program.

The Loader class uses reasonable defaults for most of the initial objects. However, you can override any initial object with some user-defined values. The main elements of the initial object table are:

- `factory` The factory used by the new process to create new kernel objects, such as threads etc. This must be a capability to an object implementing the [L4::Factory](#) protocol and defaults to the factory object provided to Ned.
- `mem` The memory allocator provided to the application and used by Ned allocates data spaces for the process. This defaults to Ned's memory allocator object (see [L4Re::Mem_alloc](#)).
- `rm_fab` The generic factory object used to allocate the region-map object for the process. (defaults to Ned's memory allocator).
- `log_fab` The generic factory to create the [L4Re::Log](#) object for the application's output (defaults to Ned's memory allocator). The `create` method of the `log_fab` object is called with `log_tag` and `log_color`, from this table, as arguments.
- `log_tag` The string used for tagging log output of this process (defaults to the program name) (see [log_fab](#)).
- `log_color` The color used for the log tag (defaults to "white").
- `scheduler` The scheduler object used for the process' threads (defaults to Ned's own scheduler).
- `caps` The table with application-specific named capabilities (default is an empty table). If the table does not contain a capability with the name 'rom', the 'rom' capability from Ned's initial caps is inserted into the table.

5.2.1.5 Access to the kernel debugger

Applications can enrich the kernel debugger with information using the API defined in [l4/sys/debugger](#). In order to do so, the developer has to assign access to the kernel debugger kernel object to the application. This can be done like this:

```
L4.default_loader:start({ caps = { jdb = L4.Env.jdb; }}, "rom/example")
```

5.2.2 Command Line Options

Ned's command line syntax is:

```
[--interactive|-i] [--cmdcap|-c CAP] [--noexit] [--execute|-e STATEMENT] <lua script> [options passed to lua s
```

Ned interprets the first non-option argument `<lua script>` as the Lua script which it should load and run. All arguments following the first non-option argument are passed as arguments to the Lua script via Lua's global `arg` table.

- Interactive Option: **interactive, i**
Start ned in interactive mode. After running the initial lua script, ned will present a prompt, where Lua statements can be typed in interactively.
Must not be used together with server mode (-c).
- Command Capability Option: **cmdcap, c**
Start ned in server mode. After running the initial Lua script, ned will accept Lua statements via IPC on the given capability (see [L4Re::Ned::Cmd_control](#)).
Must not be used together with interactive mode (-i).
- No exit Option: **noexit**
Ned does not terminate if only Return is entered at ned's prompt.
- Execute Statement Option: **execute, e**
Execute the Lua statement `STATEMENT`.

5.3 Io, the Io Server

The Io server handles all platform devices and resources such as I/O memory, ports (on x86) and interrupts, and grants access to those to clients.

Upon startup Io discovers all platform devices using available means on the system, e.g. on x86 the PCI bus is scanned and the ACPI subsystem initialised. Available I/O resource can also be configured via configuration scripts.

Io's configuration consists of two parts:

- the description of the real hardware
- the description of virtual buses

Both descriptions represent a hierarchical (tree) structure of device nodes. Where each device has a set of resources attached to it. And a device that has child devices can be considered a bus.

Hardware Description

The hardware description represents the devices that are available on the particular platform including their resource descriptions, such as MMIO regions, IO-Port regions, IRQs, bus numbers etc.

The root of the hardware devices is formed by a system bus device (accessible in the configuration via `Io.system↔_bus()`). As mentioned before, platforms that support methods for device discovery may populate the hardware description automatically, for example from ACPI. On platforms that do not have support for such methods you have to specify the hardware description by hand. A simple example for this is `x86-legacy.devs`.

Virtual Bus Description

Each Io server client is provided with its own virtual bus which it can iterate to find devices. A virtual PCI bus may be a part of this virtual bus.

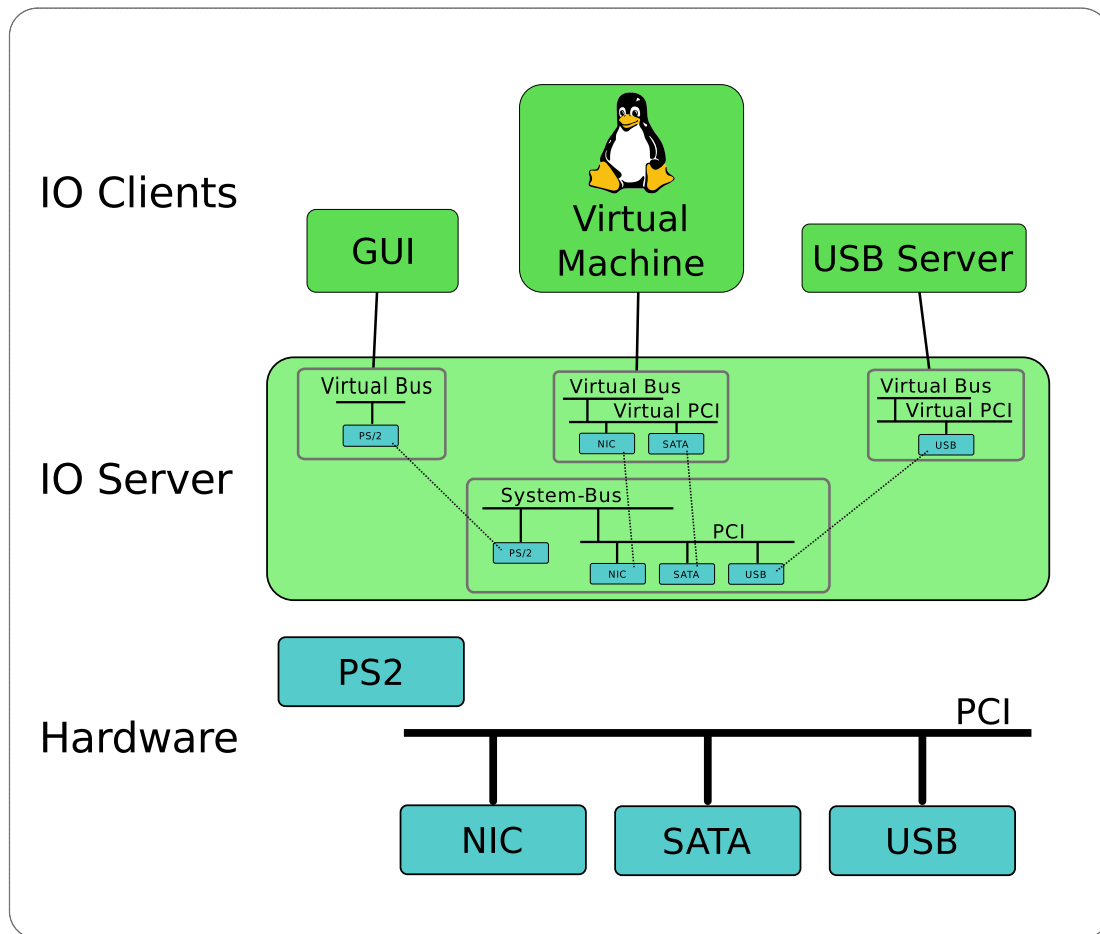


Figure 5.1 IO Service Architecture Overview

The Io server must be configured to create virtual buses for its clients.

This is done with at least one configuration file specifying static resources as well as virtual buses for clients. The configuration may be split across several configuration files passed to Io through the command line.

To allow clients access to a available devices, a virtual system bus needs to be created that lists the devices and their resources that should be available to that client. The names of the buses correspond to the capabilities given to Io in its launch configuration.

A very simple configuration for Io could look like this:

```
-- vim:ft=lua
-- Example configuration for io

-- Configure two platform devices to be known to io
Io.Dt.add_children(Io.system_bus(), function()

    -- create a new hardware device called "FOODEVICE"
    FOODEVICE = Io.Hw.Device(function()
        -- set the compatibility IDs for this device
        -- a client tries to match against these IDs and configures
        -- itself accordingly
    end)
```

```

-- the list should be sorted from specific to less specific IDs
compatible = {"dev-foo,mmio", "dev-foo"};

-- set the 'hid' property of the device, the hid can also be used
-- as a compatible ID when matching clients
Property.hid = "dev-foo,Example";

-- note: names for resources are truncated to 4 letters and a client
-- can determine the name from the ID field of a l4vbus_resource_t
-- add two resources 'irq0' and 'reg0' to the device
Resource.irq0 = Io.Res.irq(17);
Resource.reg0 = Io.Res.mmio(0x6f000000, 0x6f007fff);
end);

-- create a new hardware device called "BARDEVICE"
BARDEVICE = Io.Hw.Device(function()
  -- set the compatibility IDs for this device
  -- a client tries to match against these IDs and configures
  -- itself accordingly
  -- the list should be sorted from specific to less specific IDs
  compatible = {"dev-bar,mmio", "dev-bar"};

  -- set the 'hid' property of the device, the hid can also be used
  -- as a compatible ID when matching clients
  Property.hid = "dev-bar,Example";

  -- Specify that this device is able to use direct memory access (DMA).
  -- This is needed to allow clients to gain access to DMA addresses
  -- used by this device to directly access memory.
  Property.flags = Io.Hw_device_DF_dma_supported;

  -- note: names for resources are truncated to 4 letters and a client
  -- can determine the name from the ID field of a l4vbus_resource_t
  -- add three resources 'irq0', 'irq1', and 'reg0' to the device
  Resource.irq0 = Io.Res.irq(19);
  Resource.irq1 = Io.Res.irq(20);
  Resource.reg0 = Io.Res.mmio(0x6f100000, 0x6f100fff);
end);
end);

Io.add_vbusses
{
  -- Create a virtual bus for a client and give access to FOODEVICE
  client1 = Io.Vi.System_bus(function ()
    dev = wrap(Io.system_bus():match("FOODEVICE"));
  end);

  -- Create a virtual bus for another client and give it access to BARDEVICE
  client2 = Io.Vi.System_bus(function ()
    dev = wrap(Io.system_bus():match("BARDEVICE"));
  end);
}

```

Each device supports a 'compatible' property. It is a list of compatibility strings. A client matches itself against one (or multiple) compatibility IDs and configures itself accordingly. All other device members are handled according to their type. If the type is a resource (Io.Res) it is added as a named resource. Note that resource names are truncated to 4 letters and are stored in the ID field of a [l4vbus_resource_t](#). If the type is a device it is added as a child device to the current one. All other types are treated as a device property which can be used to configure a device driver. Right now, device properties are internal to Io only.

Matching and Assigning PCI Devices

Assigning clients PCI devices could look like this:

```

-- This is a configuration snippet for PCI device selection

local hw = Io.system_bus();

Io.add_vbusses
{
  pciclient = Io.Vi.System_bus(function ()
    PCI = Io.Vi.PCI_bus(function ()
      pci_mm      = wrap(hw:match("PCI/CC_04"));
      pci_net     = wrap(hw:match("PCI/CC_02"));
      pci_storage = wrap(hw:match("PCI/CC_01"));
    end)
  end)
}

```

The "PCI/" is followed by a bus-specific ID string. The format of the PCI ID string may be one of the following:

- PCI/CC_cc
- PCI/CC_ccss
- PCI/CC_ccssp
- PCI/VEN_vvvv
- PCI/DEV_ddd
- PCI/SUBSYS_sssssss
- PCI/REV_rr
- PCI/ADR_xxxx:xx:xx.x

Where:

- `cc` is the hexadecimal representation of the class code byte
- `ss` is the hexadecimal representation of the subclass code byte
- `pp` is the hexadecimal representation of the programming interface byte
- `vvvv` is the hexadecimal representation of the vendor ID
- `ddd` is the hexadecimal representation of the device ID
- `ssssssss` is the hexadecimal representation of the subsystem ID
- `rr` is the hexadecimal representation of the revision byte
- `xxxx:xx:xx.x` is the bus address in PCI nomenclature

As a special extension lo supports replacing the ID string with a human-readable common PCI class name. The following table gives an overview of the names known to lo and their respective PCI class and subclass.

Common Name	Description	PCI ID string
storage	Mass storage controller	CC_01
scsi	SCSI storage controller	CC_0100
ide	IDE interface	CC_0101
floppy	Floppy disk controller	CC_0102
raid	RAID bus controller	CC_0104
ata	ATA controller	CC_0105
sata	SATA controller	CC_0106
sas	Serial attached SCSI controller	CC_0107
nvm	Non-volatile memory controller	CC_0108
-	-	-
network	Network controller	CC_02
ethernet	Ethernet controller	CC_0200
token_ring	Token ring network controller	CC_0201
fddi	FDDI network controller	CC_0202
atm	ATM network controller	CC_0203
isdn	ISDN controller	CC_0204
picmg	PICMG controller	CC_0206
net_infiniband	Infiniband controller	CC_0207

Common Name	Description	PCI ID string
fabric	Fabric controller	CC_0208
network_nw	Network controller e.g. Wifi	CC_0280
-	-	-
display	Display controller	CC_03
vga	VGA compatible controller	CC_0300
xga	XGA compatible controller	CC_0301
-	-	-
media	Multimedia controller	CC_04
mm_video	Multimedia video controller	CC_0400
mm_audio	Multimedia audio controller	CC_0401
telephony	Computer telephony device	CC_0402
audio	Audio device	CC_0403
-	-	-
bridge	Bridge	CC_06
br_host	Host bridge	CC_0600
br_isa	ISA bridge	CC_0601
br_eisa	EISA bridge	CC_0602
br_microchannel	MicroChannel bridge	CC_0603
br_pci	PCI bridge	CC_0604
br_pcmcia	PCMCIA bridge	CC_0605
br_nubus	NuBus bridge	CC_0606
br_cardbus	CardBus bridge	CC_0607
br_raceway	RACEway bridge	CC_0608
br_semi_pci	Semi-transparent PCI-to-PCI bridge	CC_0609
br_infiniband_to_pci	InfiniBand to PCI host bridge	CC_060a
-	-	-
com	Communication controller	CC_07
com_serial	Serial controller	CC_0700
com_parallel	Parallel controller	CC_0701
com_multiport_ser	Multiport serial controller	CC_0702
com_modem	Modem	CC_0703
com_gpiib	GPIO controller	CC_0704
com_smart_card	Smart card controller	CC_0705
-	-	-
serial_bus	Serial bus controller	CC_0c
firewire	FireWire (IEEE 1394)	CC_0c00
access_bus	ACCESS bus	CC_0c01
ssa	SSA	CC_0c02
usb	USB controller	CC_0c03
fibre_channel	Fibre channel	CC_0c04
smbus	SMBus	CC_0c05
bus_infiniband	InfiniBand	CC_0c06
ipmi_smic	IPMI SMIC interface	CC_0c07
sercos	SERCOS interface	CC_0c08
canbus	CAN bus	CC_0c09
-	-	-
wireless	Wireless controller	CC_0d
bluetooth	Bluetooth	CC_0d11
w_8021a	802.1a controller	CC_0d20
w_8021b	802.1b controller	CC_0d21

Strong Matching of PCI Devices

If more specific matching of PCI devices is required it is possible to concatenate multiple ID strings using `&`. An example where a specific device from a specific vendor at a fixed bus address is matched would use the string `PCI/VEN_vvvv&DEV_dddd&ADR_xxxx:xx:xx.x`.

5.4 Uvmm, the virtual machine monitor

Setting up guest memory

In the most simple setup, memory for the guest can be provided via a simple dataspace. In your ned script, create a new dataspace of the required size and hand it into uvmm as the `ram` capability:

```
local ramds = L4.Env.user_factory:create(L4.Proto.Dataspace, 60 * 1024 * 1024)
L4.default_loader::startv({caps = {ram = ramds:m("rw")}}, "rom/uvmm")
```

The memory will be mapped to the most appropriate place and a memory node added to the device tree, so that the guest can find the memory.

For a more complex setup, the memory can be configured via the device tree. uvmm scans for memory nodes and tries to set up the memory from them. A memory device node should look like this:

```
memory@0 {
    device_type = "memory";
    reg = <0x00000000 0x00100000
          0x00200000 0xffffffff>;
    l4vmm,dscap = "memcap";
    l4vmm,physmap;
    dma-ranges = <>;
};
```

The `device_type` property is mandatory and needs to be set to `memory`.

`l4vmm,dscap` contains the name of the capability containing the dataspace to be used for the RAM. `reg` describe the memory regions to use for the memory. The regions will be filled up to the size of the supplied dataspace. If they are larger, then the remaining area will be cut.

`l4vmm,physmap` indicates that uvmm should try to map the dataspace to its actual physical address when no IOMMU is available. If the physical address cannot be determined or an IOMMU is available, then the memory will be mapped to the addresses supplied in `regs`. It is possible to omit the `regs` property when `l4vmm,physmap` is set. In this case, uvmm will fail to start if the physical address cannot be determined.

If a `dma-ranges` property is given, the host-physical address ranges for the memory regions will be added here. Note that the property is not cleared first, so it should be left empty.

Memory layout

uvmm populates the RAM with the following data:

- kernel binary
- (optional) ramdisk
- (optional) device tree

The kernel binary is put at the predefined address. For ELF binaries, this is an absolute physical address. If the binary supports relative addressing, the binary is put to the requested offset relative to beginning of the first 'memory' region defined in the device tree.

The ramdisk and device tree are placed as far as possible to the end of the regions defined in the first 'memory' node.

If there is a part of RAM that must remain empty, then define an extra memory node for it in the device tree. uvmm only writes to memory in the first memory node it finds.

Warning: uvmm does not touch any unpopulated memory. In particular, it does not ensure that the memory is cleared. It is the responsibility of the provider of the RAM dataspace to make sure that no data leakage can happen. Normally this is not an issue because dataspaces are guaranteed to be cleaned when they are newly created but users should be careful when reusing memory or dataspaces, for example, when restarting the uvmm.

Forwarding hardware resources to the guest

Hardware resources must be specified in two places: the device tree contains the description of all hardware devices the guest could see and the Vbus describes which resources are actually available to the uvmm.

The vbus allows the uvmm access to hardware resources in the same way as any other [L4](#) application. uvmm expects a capability named 'vbus' where it can access its hardware resources. It is possible to leave out the capability for purely virtual guests (Note that this is not actually practical on some architectures. On ARM, for example, the guest needs hardware access to the interrupt controller. Without a 'vbus' capability, interrupts will not work.) For information on how to configure a vbus, see the [IO documentation](#).

The device tree needs to contain the hardware description the guest should see. For hardware devices this usually means to use a device tree that would also be used when running the guest directly on hardware.

On startup, uvmm scans the device tree for any devices that require memory or interrupt resources and compares the required resources with the ones available from its vbus. When all resources are available, it sets up the appropriate forwarding, so that the guest now has direct access to the hardware. If the resources are not available, the device will be marked as 'disabled'. This mechanism allows to work with a standard device tree for all guests in the system while handling the actual resource allocation in a flexible manner via the vbus configuration.

The default mechanism assigns all resources 1:1, i.e. with the same memory address and interrupt number as on hardware. It is also possible to map a hardware device to a different location. In this case, the assignment between vbus device and device tree device must be known in advance and marked in the device tree using the `l4vmm, vbus-dev` property.

The following device will for example be bound with the vbus device with the HID 'l4-test,dev':

```
test@e0000000 {
    compatible = "memdev,bar";
    reg = <0 0xe0000000 0 0x50000>,
        <0 0xe1000000 0 0x50000>;
    l4vmm,vbus-dev = "l4-test,dev";
    interrupts-extended = <&gic 0 139 4>;
};
```

Resources are then matched by name. Memory resources in the vbus must be named `reg0` to `reg9` to match against the address ranges in the device tree `reg` property. Interrupts must be called `irq0` to `irq9` and will be matched against `interrupts` or `interrupts-extended` entries in the device tree. The vbus must expose resources for all resources defined in the device tree entry or the initialisation will fail.

An appropriate IO entry for the above device would thus be: `MEM = Io.Hw.Device(function() Property.hid = "l4-test,dev" Resource.reg0 = Io.Res.mmio(0x41000000, 0x4104ffff) Resource.reg1 = Io.Res.mmio(0x42000000, 0x4204ffff) Resource.irq0 = Io.Res.irq(134); end)`

Please note that HIDs on the vbus are not necessarily unique. If multiple devices with the HID given in `l4vmm, vbus-dev` are available on the vbus, then one device is chosen at random.

If no vbus device with the given HID is available, the device is disabled.

How to enable guest suspend/resume

Note

Currently only supported on ARM. It should work fine with Linux version 4.4 or newer.

Uvmm (partially) implements the power state coordination interface (PSCI), which is the standard ARM power management interface. To make use of this interface, you have to announce its availability to the guest operating system via the device tree like so:

```
psci {
    compatible = "arm,psci-0.2";
    method = "hvc";
};
```

The Linux guest must be configured with at least these options:

```
CONFIG_SUSPEND=y
CONFIG_ARM_PSCI=y
```

How to communicate power management (PM) events

Uvmm can be instructed to inform a PM manager of PM events through the [L4::Platform_control](#) interface. To that end, uvmm may be equipped with a `pfc` cap. On suspend, uvmm will call `l4_platform_ctl_system_suspend()`.

The `pfc` cap can also be implemented by IO. In that case the guest can start a machine suspend/shutdown/reboot.

KVM clock for uvmm/amd64 guests

When executing [L4Re](#) + uvmm on QEMU, the PIT as clock source is not reliable. The paravirtualized KVM clock provides the guest with a stable clock source.

A KVM clock device is available to the guest, if the device tree contains the corresponding entry:

```
kvm_clock {
    compatible = "kvm-clock";
    reg = <0x0 0x0 0x0 0x0>;
};
```

To make use of this clock, the Linux guest must be built with the following configuration options:

CONFIG_HYPERVISOR_GUEST=y CONFIG_KVM_GUEST=y CONFIG_PTP_1588_CLOCK_KVM is not set

Note: KVM calls besides the KVM clock are unhandled and lead to failure in the uvmm, e.g. vmcall 0x9 for the PTP_1588_CLOCK_KVM.

This is considered a development feature. The KVM clock is not required when running on physical hardware as TSC calibration via the PIT works as expected.

5.5 Mag, the GUI Multiplexer

Mag is the default multiplexer for graphics hardware.

It is not, and does not attempt to be, a fully-fledged window manager.

Command Line Options

As Mag's only command line option it supports loading additional plugins via the application's command line. Plugins must be either a Lua file or a shared library. Shared libraries must be named `libmag-$LIBNAME.so`.

Mag Sessions

Mag provides two types of sessions which a client can create via the Factory interface.

1. Mag client session

A client with a mag client session gets access to the whole screen. The client has to allocate and manage its own buffers and has to position them on the screen on its own. Mag provides the factory to create client sessions via the capability named `mag`.

2. Client framebuffer session

For a client framebuffer session mag allocates a view of the requested size and displays it at the requested coordinates on the screen. Mag provides the factory to create framebuffer sessions via the capability named `SVC`.

The options described below are options the client provides to the [L4::Factory::create\(\)](#) call. These options influence the appearance and behaviour of the newly created session.

Session Factory Options

As a simple nitpicker clone Mag supports the so-called Xray mode. This mode displays all session labels and draws a colored frame around them. The session that currently has the input focus is highlighted. The Xray mode is activated via the special keys *Scroll* or *NEXTSONG*.

Mag allows to define a text label and a color for all client session types. The label and the color are displayed when Mag enters the Xray mode.

- Label Option: **label**

`l=LABEL, label=LABEL` Set the session's text label to LABEL. The label is restricted to a length of 256 characters.

- Color Option: **col**

`col=COLOR` Set the session's color which is used in Xray mode to tint the session's screen area and the border drawn around it. The argument can be either one of the following letters or a hexadecimal representation of the RGB values.

- `r`, `R` Red color
- `g`, `G` Green color
- `b`, `B` Blue color
- `w`, `W` White color
- `y`, `Y` Yellow color
- `v`, `V` Magenta color

Example

```
-- set label to "Linux" and use a light blue color
fb = mag_client:create(L4.Proto.Goos, "l=Linux", "col=98d9ff");
```

Mag Client Session Options

These options only apply to Mag client sessions.

- Default Background Option: **default-background**

`df1-bg, default-background` Marks this session as the default background.

Mag Client Framebuffer Session Options

These options only apply to Mag client framebuffer sessions.

- Geometry Option: **geometry**

`g=GEOMETRY, geometry=GEOMETRY` Set the session's geometry and position on the screen. GEOMETRY is provided in an X11-style format, e.g. `g=WIDTHxHEIGHT+X_OFFSET+Y_OFFSET`.

- Focus Option: **focus**

`focus` Set the focus to this session.

- Collapsed Option: **shaded**

`shaded` The window is collapsed and only the title bar is visible. The window can be expanded by clicking into the title bar with the middle mouse button. Collapsing and expanding works also independently of this option.

- Fixed Option: **fixed** `fixed` The window cannot be moved on the screen.

- Barheight Option: **barheight**

`barheight=X` Set the height of the title bar in pixels.

Example

```
-- create a window of 640x480 pixels at position (100,100) on the screen.
fb = mag_fb:create(L4.Proto.Goos, "g=640x480+100+100");
```

5.6 Cons, the Console Multiplexer

`cons` allows to multiplex console output from different [L4](#) clients to different [L4::Vcon](#)-capable in/output servers.

Multiplexers and Frontends

`cons` is able to connect multiple clients with multiple in/output servers.

Clients are handled by a *multiplexer*. Each multiplexer publishes a server capability that allows to create new client connections. The default multiplexer is normally known under the `cons` capability.

Actual in/output is handled by separate frontends. From the point-of-view of `cons`, a frontend consists of an IPC channel to a server that speaks an appropriate server protocol. By default the `L4.Env.log` capability is used.

At the moment, `cons` only implements the [L4::Vcon](#) protocol for both, clients and frontends.

Command Line Options

`cons` accepts the following command line switches:

- `-a, --show-all`
Initially show output from all clients.
- `-c <client>, --autoconnect <client>`
Automatically connect to the client with the given name. That means that output of this client will be visible and input will be routed to it.
- `-k, --keep`
Keep the console buffer when a client disconnects.
- `-n, --defaultname`
Default multiplexer capability to use. Default: `cons`.
- `-B <size>, --defaultbufsize <size>`
Default buffer size per client in bytes. Default: 40960
- `-m <prompt name>, --mux <prompt name>`
Add a new multiplexer named `<prompt name>`. This is necessary if output should be sent to different frontends.
- `-f <cap>, --frontend <cap>`
Set the frontend for the current multiplexer. Output for the multiplexer is then sent to the capability with the given name. The server connected to the capability needs to understand the [L4::Vcon](#) protocol.

Chapter 6

Deprecated List

Global **L4::Factory::create_irq** (Cap< Irq >const &target_cap, l4_utcb_t *utcb=**l4_utcb()**)

Use **create()** with **Cap<Irq>** as argument instead.

Global **L4::Factory::create_thread** (Cap< Thread > const &target_cap, l4_utcb_t *utcb=**l4_utcb()**)

Use **create()** with **Cap<Thread>** as argument instead.

Global **L4::Factory::create_vm** (Cap< Vm >const &target_cap, l4_utcb_t *utcb=**l4_utcb()**)

Use **create()** with **Cap<Vm>** as argument instead.

Class **L4::lpc_svr::Timed_work< HOOKS >**

Use **L4::lpc_svr::Timeout_queue_hooks**

Global **L4::lirq::attach** (l4_umword_t label, Cap< Thread > const &thread=**Cap< Thread >::Invalid**, l4_utcb_t *utcb=**l4_utcb()**)

Use **bind_thread()**.

Global **L4::lirq::unmask** (l4_utcb_t *utcb=**l4_utcb()**)

Use **L4::lirq_eoi::unmask()**

Global **L4::Task::cap_has_child** (Cap< void > const &cap, l4_utcb_t *utcb=**l4_utcb()**)

Do not use. Undetermined future, might be removed.

Global **l4_task_cap_has_child** (l4_cap_idx_t task, l4_cap_idx_t cap) **L4_NOTHROW**

Do not use. Undetermined future, might be removed.

Global **L4Re::Dataspace::phys** (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size)

Use **L4Re::Dma_space** instead. This function will be removed in future releases.

Global **L4Re::Mem_alloc::free** (L4::Cap< Dataspace > mem) const

This function is an empty stub which remains here for backward compatibility only. Use **L4::Task::unmap(mem, L4_FP_DELETE_OBJ)** or other similar means to remove a dataspace.

Class **L4Re::Util::Auto_cap< T >**

Use **L4Re::Util::Unique_cap**.

Class **L4Re::Util::Auto_del_cap< T >**

Use **L4Re::Util::Unique_cap**.

Global **L4Re::Util::make_auto_cap** ()

Use **L4Re::Util::make_unique_cap()**.

Global **L4Re::Util::make_auto_del_cap** ()

Use **L4Re::Util::make_unique_del_cap()**.

Global **l4re_ma_free** (l4re_ds_t const mem) **L4_NOTHROW**

This function is deprecated. Use **l4_task_unmap()** or similar means to remove a dataspace.

Global [l4re_ma_free_srv](#) ([l4_cap_idx_t](#) srv, [l4re_ds_t](#) const mem) [L4_NOTHROW](#)

This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Global [L4virtio::Device::register_iface](#) ([L4::lpc::Cap](#)< [L4::Triggerable](#) > guest_irq, [L4::lpc::Out](#)< [L4::Cap](#)< [L4::Triggerable](#) > > host_irq, [L4::lpc::Out](#)< [L4::Cap](#)< [L4Re::Dataspace](#) > > config_ds)

Use [device_config\(\)](#), [device_notification_irq\(\)](#) and [l4u::bind\(\)](#) instead.

Chapter 7

Module Index

7.1 Modules

Here is a list of all modules:

Base API	103
Basic Macros	106
C++ IPC Interface Definition.	120
Internal Helpers	306
Cache Consistency	124
Capabilities	128
Error codes	200
Fiasco extensions	225
Fiasco real time scheduling extensions	231
Kernel Debugger	324
Flex pages	234
Integer Types	290
Kernel Interface Page	327
Fiasco-UX Virtual devices	232
Memory descriptors (C version)	413
Kernel Objects	335
Factory	212
IPC-Gate API	266
IRQs	270
Interrupt controller	310
L4 kernel object type information	367
Platform Control C API	468
Scheduler	491
Task	519
Thread	527
Thread control	545
vCPU API	623
Virtual Console	604
Virtual Machines	617
VM API for SVM	575
VM API for TZ	576
VM API for VMX	577
Memory operations.	418
Memory related	421

Object Invocation	454
Error Handling	193
Message Items	428
Message Tag	432
Realtime API	477
Timeouts	550
Virtual Registers (UTCBs)	618
ARM Virtual Registers (UTCB)	91
Buffer Registers (BRs)	117
Message Registers (MRs)	431
Exception registers	207
Thread Control Registers (TCRs)	544
amd64 Virtual Registers (UTCB)	622
x86 Virtual Registers (UTCB)	634
DMA API for L4Re	149
EDID parsing functionality	161
IO interface	258
IRQ handling library	269
Interface for asynchronous ISR handlers.	295
Interface for asynchronous ISR handlers with a given IRQ capability.	293
Interface using direct functionality.	297
Interface using direct functionality.	303
L4 IPC Opcodes	340
L4 VIRTIO Interface	346
L4 VIRTIO Block Device	344
L4 VIRTIO Transport Layer	347
L4 Vbus functions	357
L4vbus GPIO functions	383
L4vbus PCI functions	394
L4Re C Interface	369
Capability allocator	132
DMA Space Interface	150
Dataspace interface	154
Debug interface	159
Event interface	203
Initial Environment	285
Kumem allocator utility	337
L4Re Util C Interface	381
Log interface	400
Memory allocator	407
Namespace interface	449
Region map interface	479
Video API	594
L4Re C++ Interface	372
Auxiliary data	102
C++ Exceptions	119
Console API	140
Debugging API	160
Event API	202
L4Re ELF Auxiliary Information	377
L4Re Protocol identifiers	380
L4Re Util C++ Interface	382
Kumem utilities	338
L4Re Capability API	374
Logging interface	404
Name-space API	448

Parent API	467
Region map API	478
Vbus API	593
Server-Side IPC framework	497
Shared Memory Library	499
Chunks	133
Consumer	141
Producer	472
Signals	511
Consumer	145
Producer	471
Sigma0 API	505
Internal constants	307
Small C++ Template Library	515
Utility Functions	568
Atomic Instructions	92
Bit Manipulation	109
CPU related functions	121
Comfortable Command Line Parsing	137
ELF binary format	166
Functions to manipulate the local IDT	250
IA32 Port I/O API	251
Internal functions	309
Kernel Interface Page API	332
Low-Level Thread Functions	405
Machine Restarting Function	406
Random number support	475
Timestamp Counter	559
vCPU Support Library	626
Extended vCPU support	210

Chapter 8

Namespace Index

8.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

cxx	Our C++ library	637
cxx::Bits	Internal helpers for the cxx package	639
L4	L4 low-level kernel interface	640
L4::lpc	IPC related functionality	651
L4::lpc::Msg	IPC Message related functionality	660
L4::lpc_svr	Helper classes for L4::Server instantiation	666
L4::Typeid	Definition of interface data-type helpers	667
L4::Types	L4 basic type helpers for C++	667
L4Re	L4Re C++ Interfaces	668
L4Re::Util	Documentation of the L4 Runtime Environment utility functionality in C++	682
L4Re::Vfs	Virtual file system for interfaces POSIX libc	692
L4virtio	L4-VIRTIO Transport C++ API	693

Chapter 9

Hierarchical Index

9.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

L4::lpc::Array_ref< CHAR, LEN >	954
L4::lpc::Array< CHAR, LEN >	950
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, Bits::Avl_set_get_key< ITEM_TYPE > > .	756
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	704
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >	756
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	700
cxx::Bits::Base_avl_set< Pair< Region, Hdlr >, cxx::Lt_functor< Region >, Alloc, Bits::Avl_map_get_← key< Region > >	756
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >	700
cxx::Base_slab< sizeof(Type), Slab_size, Max_free, Alloc >	715
cxx::Slab< Type, Slab_size, Max_free, Alloc >	839
cxx::Base_slab_static< sizeof(Type), Slab_size, Max_free, Alloc >	718
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	843
cxx::Bits::Basic_list< Bits::Basic_list_policy< Observer, H_list_item > >	769
cxx::H_list< Observer >	792
cxx::Bits::Basic_list< Bits::Basic_list_policy< Slab_i, H_list_item > >	769
cxx::H_list< Slab_i >	792
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >	769
cxx::H_list< T >	792
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > >	769
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >	792
cxx::H_list_t< T >	801
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >	769
cxx::S_list< T >	836
cxx::Bits::Basic_list< Bits::Basic_list_policy< Timeout, H_list_item > >	769
cxx::H_list< Timeout >	792
cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > .	769
cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_← base > > >	792
cxx::H_list_t< Weak_ref_base >	801

L4::Types::Bool< __iface_conflict< I, I2 >::value _iface_conflict< I, LIST::type >::value >	1217
L4::Types::Bool< false >	1217
L4::Types::False	1218
L4::Types::Same< A, B >	1225
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto >	1217
L4::Types::Bool< _iface_conflict< I, L2::type >::value _Conflict< L1::type, L2::type >::value >	1217
L4::Types::Bool< true >	1217
L4::Types::True	1227
L4::ipc::Msg::Is_valid_rpc_type< A * >	1002
L4::ipc::Msg::Is_valid_rpc_type< T >	1002
L4::Types::Bool< Typeid::Conflict< L1::type, L2::type >::value Conflict< L1, LIST... >::value Conflict< L2, LIST... >::value >	1217
L4::Types::Bool< Typeid::Iface_conflict< I::type, L::type >::value Iface_conflict< I, LIST... >::value >	1217
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< KEY_TYPE >, COMPARE< KEY_TYPE > >	771
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, COMPARE< KEY_TYPE > >	707
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > >	771
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > >	707
cxx::Bits::Bst< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, COMPARE >	771
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, COMPARE >	707
cxx::Bits::Bst< _Node, GET_KEY, COMPARE >	771
cxx::Avl_tree< _Node, GET_KEY, COMPARE >	707
cxx::Bits::Bst< Entry, Names_get_key, Lt_functor< typename Names_get_key ::Key_type > >	771
cxx::Avl_tree< Entry, Names_get_key >	707
L4::ipc::Msg::Cnt_val_ops< Detail::_Plain< T >::type, DIR, CLASS >	986
L4::ipc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< A * > ::arg_type >::type, typename Direction< A * >::type, typename Class< typename Detail::_Plain< A * >::type >::type >	986
L4::ipc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< A const * > ::arg_type >::type, typename Direction< A const * >::type, typename Class< typename Detail::_Plain< A const * >::type >::type >	986
L4::ipc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< Array< A, LEN > & > ::arg_type >::type, typename Direction< Array< A, LEN > & >::type, typename Class< typename Detail::_Plain< Array< A, LEN > & >::type >::type >	986
L4::ipc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< Array< A, LEN > > ::arg_type >::type, typename Direction< Array< A, LEN > >::type, typename Class< typename Detail::_Plain< Array< A, LEN > >::type >::type >	986
L4::ipc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< String< A, LEN > & > ::arg_type >::type, typename Direction< String< A, LEN > & >::type, typename Class< typename Detail::_Plain< String< A, LEN > & >::type >::type >	986
L4::ipc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< T > ::arg_type >::type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type >	986
cxx::Auto_ptr< T >	695
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	715
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	718
cxx::Bitfield< T, LSB, MSB >	723
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	734
cxx::Bitfield< T, LSB, MSB >::Value< TT >	733
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	736
cxx::Bitmap_base	741
cxx::Bitmap< BITS >	737
cxx::Bitmap_base::Bit	752
cxx::Bitmap_base::Char< BITS >	753
cxx::Bitmap_base::Word< BITS >	753
cxx::Bits::Avl_map_get_key< KEY_TYPE >	755
cxx::Bits::Avl_set_get_key< KEY_TYPE >	755
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	756
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	767

cxx::Bits::Basic_list< POLICY >	769
cxx::H_list< T, POLICY >	792
cxx::S_list< T, POLICY >	836
cxx::Bits::Bst< Node, Get_key, Compare >	771
cxx::Avl_tree< Node, Get_key, Compare >	707
cxx::Bits::Bst_node	785
cxx::Avl_tree_node	712
cxx::Bits::Direction	787
cxx::Bits::Smart_ptr_list< ITEM >	789
cxx::Bits::Smart_ptr_list_item< T, STORE_T >	791
cxx::Ref_obj_list_item< T >	829
cxx::H_list_item_t< ELEM_TYPE >	799
L4::lpc_svr::Timeout	1053
cxx::List< D, Alloc >	804
cxx::List< D, Alloc >::Iter	807
cxx::List_alloc	808
cxx::List_item	812
cxx::List_item::Iter	818
cxx::List_item::T_iter< T, Poly >	821
cxx::List_item::T_iter< E >	821
cxx::Lt_functor< Obj >	824
cxx::New_allocator< _Type >	824
cxx::Nothrow	825
cxx::Pair< First, Second >	826
cxx::Pair_first_compare< Cmp, Typ >	827
cxx::Ref_ptr< T, CNT >	830
cxx::static_vector< T, IDX >	846
cxx::String	848
L4Re::Util::Names::Name	1416
L4virtio::Svr::Device_t< Ds_data >	1646
L4virtio::Svr::Block_dev< Ds_data >	1610
L4virtio::Svr::Driver_mem_list_t< Ds_data >	1652
L4virtio::Svr::Driver_mem_region_t< Ds_data >	1659
Elf32_Dyn	860
Elf32_Ehdr	861
Elf32_Phdr	863
Elf32_Shdr	864
Elf32_Sym	865
Elf64_Dyn	866
Elf64_Ehdr	867
Elf64_Phdr	869
Elf64_Shdr	870
Elf64_Sym	871
cxx::H_list_item_t< Weak_ref_base >	799
cxx::Weak_ref_base	857
cxx::Weak_ref< T >	855
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >	1092
L4Re::Console	1289
L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY >	1092
L4::Kobject_x< Iommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > >	1106
L4::Iommu	946
L4::Alloc_list	872
L4::Arm_smccc	873
L4::Basic_registry	876
L4Re::Util::Object_registry	1421

L4::Cap_base	886
L4::Cap< L4::Factory >	881
L4::Cap< L4::Irq >	881
L4::Cap< L4::Semaphore >	881
L4::Cap< L4::Thread >	881
L4::Cap< L4::Vcon >	881
L4::Cap< L4Re::Dataspace >	881
L4::Cap< L4Re::Namespace >	881
L4::Cap< L4Re::Rm >	881
L4::Cap< L4Re::Video::Goos >	881
L4::Cap< L4vbus::Vbus >	881
L4::Cap< void >	881
L4::Cap< T >	881
L4::Smart_cap< T, SMART >	1160
L4::Exception	910
L4::Exception_tracer	911
L4::Base_exception	874
L4::Invalid_capability	940
L4::Runtime_error	1136
L4::Bounds_error	878
L4::Com_error	897
L4::Element_already_exists	906
L4::Element_not_found	908
L4::Out_of_memory	1111
L4::Unknown_error	1229
L4::Factory::Lstr	924
L4::Factory::Nil	925
L4::Factory::S	926
L4::Io_pager	944
L4::Kobject_t< Pager, Io_pager, L4_PROTO_PAGE_FAULT >	1100
L4::Pager	1114
L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >	1100
L4Re::Rm	1357
L4::IOModifier	949
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	953
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	954
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	950
L4::lpc::As_value< T >	956
L4::lpc::Buf_item	957
L4::lpc::Call	958
L4::lpc::Call_t< RIGHTS >	959
L4::lpc::Call_zero_send_timeout	961
L4::lpc::Cap< T >	962
L4::lpc::Gen_fpage< T >	965
L4::lpc::In_out< T >	969
L4::lpc::Istream	976
L4::lpc::Iostream	969
L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >	986
L4::lpc::Msg::Cls_buffer	987
L4::lpc::Msg::Do_rcv_buffers	997
L4::lpc::Msg::Cls_data	989
L4::lpc::Msg::Do_in_data	993
L4::lpc::Msg::Do_out_data	995
L4::lpc::Msg::Cls_item	990
L4::lpc::Msg::Do_in_items	994

L4::lpc::Msg::Do_out_items	996
L4::lpc::Msg::Dir_in	991
L4::lpc::Msg::Do_in_data	993
L4::lpc::Msg::Do_in_items	994
L4::lpc::Msg::Do_rcv_buffers	997
L4::lpc::Msg::Dir_out	992
L4::lpc::Msg::Do_out_data	995
L4::lpc::Msg::Do_out_items	996
L4::lpc::Msg::Elem< Array< A, LEN > &>	999
L4::lpc::Msg::Elem< Array< A, LEN > >	1000
L4::lpc::Msg::Elem< Array_ref< A, LEN > &>	1001
L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	1004
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	1004
L4::lpc::Msg_ptr< T >	1005
L4::lpc::Opt< T >	1006
L4::lpc::Ostream	1008
L4::lpc::lostream	969
L4::lpc::Out< T >	1014
L4::lpc::Ret_array< T >	1015
L4::lpc::Send_only	1016
L4::lpc::Small_buf	1016
L4::lpc::Snd_item	1018
L4::lpc::Str_cp_in< T >	1018
L4::lpc::Varg	1020
L4::lpc::Varg_list_ref	1026
L4::lpc::Varg_list< MAX >	1025
L4::lpc_svr::Compound_reply	1035
L4::lpc_svr::Default_loop_hooks	1036
L4Re::Util::Br_manager_hooks	1379
L4::lpc_svr::Default_setup_wait	1037
L4::lpc_svr::Default_timeout	1038
L4::lpc_svr::Default_loop_hooks	1036
L4Re::Util::Br_manager_hooks	1379
L4::lpc_svr::Direct_dispatch< R >	1039
L4::lpc_svr::Exc_dispatch< R, Exc >	1042
L4::lpc_svr::Direct_dispatch< R * >	1041
L4::lpc_svr::Ignore_errors	1043
L4::lpc_svr::Default_loop_hooks	1036
L4Re::Util::Br_manager_hooks	1379
L4Re::Util::Br_manager_timeout_hooks	1380
L4::lpc_svr::Server_iface	1044
L4::lpc_svr::Br_manager_no_buffers	1031
L4::lpc_svr::Default_loop_hooks	1036
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1060
L4Re::Util::Br_manager	1374
L4Re::Util::Br_manager_hooks	1379
L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >	1060
L4Re::Util::Br_manager_timeout_hooks	1380
L4::lpc_svr::Timed_work< HOOKS >	1053
L4::lpc_svr::Timeout_queue	1056
L4::lpc::Irq_eoi	1070
L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >	1100
L4::lcu	931
L4::Kobject_t< Device, L4::lcu, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > >	1100
L4virtio::Device	1593

L4::Kobject_t< Event, L4::Icu, L4RE_PROTO_EVENT >1100
L4Re::Event1321
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >1096
L4vbus::Vbus1569
L4::Kobject_t< Scheduler, Icu, L4_PROTO_SCHEDULER, Type_info::Demand_t< 1 > >1100
L4::Scheduler1139
L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG >1100
L4::Vcon1232
L4::Kobject_t< Log, L4::Vcon, L4::PROTO_EMPTY >1100
L4Re::Log1334
L4::Kobject_t< Triggerable, Irq_eoi, L4_PROTO_IRQ >1100
L4::Triggerable1188
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >1092
L4::Irq1063
L4::Kobject_t< Irq_mux, Triggerable, L4_PROTO_IRQ_MUX >1100
L4::Irq_mux1076
L4::Kobject_t< Semaphore, Triggerable, L4_PROTO_SEMAPHORE >1100
L4::Semaphore1144
L4::Kip::Mem_desc1079
L4::Kobject1088
L4::Kobject_t< Arm_smccc, L4::Kobject, PROTO, Type_info::Demand_t<> >1100
L4::Kobject_t< Cmd_control, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >1100
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >1100
L4Re::Dataspace1291
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >1096
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >1100
L4Re::Debug_obj1301
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >1100
L4::Debugger899
L4::Kobject_t< Derived, L4::Kobject, PROTO, S_DEMAND >1100
L4::Kobject_t< Dma_space, L4::Kobject, PROTO, L4::Type_info::Demand_t< 1 > >1100
L4::Kobject_t< Exception, L4::Kobject, PROTO, Type_info::Demand_t<> >1100
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >1100
L4::Factory913
L4::Kobject_t< Mem_alloc, L4::Factory, L4::PROTO_EMPTY >1100
L4Re::Mem_alloc1337
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS >1100
L4Re::Video::Goos1482
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >1100
L4Re::Inhibitor1329
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >1096
L4::Kobject_t< Io_pager, L4::Kobject, PROTO, Type_info::Demand_t<> >1100
L4::Kobject_t< Irq_eoi, L4::Kobject, PROTO, Type_info::Demand_t<> >1100
L4::Kobject_t< Meta, Kobject, L4_PROTO_META >1100
L4::Meta1107
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE >1100
L4Re::Mmio_space1342
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > >1100
L4Re::Namespace1347
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT >1100
L4Re::Parent1355
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL >1100
L4::Platform_control1118

L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > . . .	1100
L4::Rcv_endpoint	1127
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >	1092
L4::Kobject_t< lpc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1100
L4::lpc_gate	1028
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >	1100
L4::Task	1164
L4::Kobject_t< Vm, Task, L4_PROTO_VM >	1100
L4::Vm	1239
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD, Type_info::Demand_t< 1 > >	1100
L4::Thread	1173
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >	1092
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >	1096
L4::Kobject_demand< T >	1099
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	1100
L4::Kobject_typeid< T >	1102
L4::Kobject_typeid< void >	1105
L4::Kobject_x< Derived, ARGS >	1106
L4::Poll_timeout_kipclock	1122
L4::Proto_t< P >	1125
L4::Registry_iface	1131
L4Re::Util::Object_registry	1421
L4::Server< LOOP_HOOKS >	1149
L4Re::Util::Registry_server< LOOP_HOOKS >	1429
L4::Server_object	1152
L4::Server_object_x< Derived, IFACE, BASE >	1158
L4::Server_object_t< Kobject >	1154
L4::Irq_handler_object	1073
L4::Server_object_t< IFACE, BASE >	1154
L4::Server_object_x< Derived, IFACE, BASE >	1158
L4::String	1164
L4::Thread::Attr	1183
L4::Thread::Modify_senders	1186
L4::Type_info	1192
L4::Type_info::Demand	1193
L4::Type_info::Demand_t< __l::Max< D1::Caps, D2::Caps >::Res, D1::Flags D2::Flags, __l::Max< D1::Mem, D2::Mem >::Res, __l::Max< D1::Ports, D2::Ports >::Res >	1196
L4::Type_info::Demand_union_t< D1, D2 >	1198
L4::Type_info::Demand_t< __l::Max< Kobject_typeid< T1 >::Demand::Caps, Kobject_demand< T2... >::Caps >::Res, Kobject_typeid< T1 >::Demand::Flags Kobject_demand< T2... >::Flags, __l::Max< Kobject_typeid< T1 >::Demand::Mem, Kobject_demand< T2... >::Mem >::Res, __l::Max< Kobject_typeid< T1 >::Demand::Ports, Kobject_demand< T2... >::Ports >::Res >	1196
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... > >	1198
L4::Type_info::Demand_t<>	1196
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	1196
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R >::Rpc< Y >	1202
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	1203
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	1204
L4::Typeid::Detail::Rpc_end	1205
L4::Typeid::Detail::_Rpc< L4::Opcode, 0, RPCS... >	1201
L4::Typeid::Rpc< RPCS >	1210
L4::Typeid::Detail::_Rpc< l4_umword_t, 0, ARG... >	1201
L4::Typeid::Rpc_sys< ARG >	1215
L4::Typeid::Detail::_Rpc< OPCODE_TYPE, 0, RPCS... >	1201

L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >	1213
L4::Typeid::Detail::_Rpc< void, 0, OPERATION >	1201
L4::Typeid::Rpc_nocode< OPERATION >	1207
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	1201
L4::Typeid::P_dispatch< LIST >	1206
L4::Typeid::Raw_ipc< CLASS >	1207
L4::Typeid::Rpc_code< OPCODE_TYPE >	1212
L4::Types::Bool< V >	1217
L4::Types::Flags< BITS_ENUM, UNDERLYING >	1220
l4_buf_regs_t	1242
l4_exc_regs_t	1243
l4_fpage_t	1246
l4_icu_info_t	1246
L4::Icu::Info	939
l4_icu_msi_info_t	1248
l4_kernel_info_mem_desc_t	1249
l4_kernel_info_t	1250
l4_msg_regs_t	1252
l4_msgtag_t	1253
l4_sched_cpu_set_t	1255
l4_sched_param_t	1258
l4_snd_fpage_t	1259
l4_thread_regs_t	1260
l4_timeout_s	1261
l4_timeout_t	1262
l4_tracebuffer_status_t	1263
l4_tracebuffer_status_window_t	1265
l4_vcon_attr_t	1266
l4_vcpu_ipc_regs_t	1267
l4_vcpu_regs_t	1268
l4_vcpu_state_t	1272
L4vcpu::Vcpu	1580
l4_vhw_descriptor	1274
l4_vhw_entry	1277
l4_vm_svm_vmcb_control_area	1280
l4_vm_svm_vmcb_state_save_area	1280
l4_vm_svm_vmcb_state_save_area_seg	1282
l4_vm_svm_vmcb_t	1282
l4_vm_tz_state	1284
L4Re::Cap_alloc	1284
L4Re::Dataspace::Stats	1300
L4Re::Dma_space	1303
L4Re::Env	1308
L4Re::Event_buffer_t< PAYLOAD >	1325
L4Re::Util::Event_buffer_t< PAYLOAD >	1405
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	1401
L4Re::Event_buffer_t< PAYLOAD >::Event	1328
L4Re::Ned::Cmd_control	1353
L4Re::Smart_cap_auto< Unmap_flags >	1370
L4Re::Smart_count_cap< Unmap_flags >	1371
L4Re::Util::Auto_cap< T >	1372
L4Re::Util::Auto_del_cap< T >	1373
L4Re::Util::Cap_alloc_base	1383
L4Re::Util::Counter< COUNTER >	1384
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	1384
L4Re::Util::Dataspace_svr	1391
L4Re::Util::Event_svr< SVR >	1409

L4Re::Util::Event_t< PAYLOAD >	1411
L4Re::Util::Item_alloc_base	1415
L4Re::Util::Names::Name_space	1417
L4Re::Util::Ref_cap< T >	1427
L4Re::Util::Ref_del_cap< T >	1428
L4Re::Util::Smart_cap_auto< Unmap_flags >	1433
L4Re::Util::Smart_count_cap< Unmap_flags >	1434
L4Re::Util::Vcon_svr< SVR >	1435
L4Re::Util::Video::Goos_svr	1436
L4Re::Vfs::Directory	1447
L4Re::Vfs::File	1452
L4Re::Vfs::Be_file	1440
L4Re::Vfs::File_system	1454
L4Re::Vfs::Be_file_system	1444
L4Re::Vfs::Fs	1456
L4Re::Vfs::Ops	1467
L4Re::Vfs::Generic_file	1461
L4Re::Vfs::File	1452
L4Re::Vfs::Mman	1466
L4Re::Vfs::Ops	1467
L4Re::Vfs::Regular_file	1470
L4Re::Vfs::File	1452
L4Re::Vfs::Special_file	1475
L4Re::Vfs::File	1452
L4Re::Video::Color_component	1478
L4Re::Video::Goos::Info	1488
L4Re::Video::Pixel_info	1490
L4Re::Video::View	1497
L4Re::Video::View::Info	1502
l4re_aux_t	1505
l4re_ds_stats_t	1506
l4re_elf_aux_mword_t	1506
l4re_elf_aux_t	1507
l4re_elf_aux_vma_t	1508
l4re_env_cap_entry_t	1509
l4re_env_t	1511
l4re_event_t	1512
l4re_video_color_component_t	1513
l4re_video_goos_info_t	1514
l4re_video_pixel_info_t	1515
l4re_video_view_info_t	1516
l4re_video_view_t	1518
l4util_idt_desc_t	1519
l4util_idt_header_t	1520
l4util_mb_addr_range_t	1521
l4util_mb_apm_t	1522
l4util_mb_drive_t	1522
l4util_mb_info_t	1524
l4util_mb_mod_t	1526
l4util_mb_vbe_ctrl_t	1527
l4util_mb_vbe_mode_t	1528
L4vbus::Gpio_module::Pin_slice	1546
L4vbus::Pm< DEC >	1568
l4vbus_device_t	1576
l4vbus_resource_t	1577
L4vcpu::State	1578

L4virtio::Ptr< T >	1604
L4virtio::Svr::Bad_descriptor	1608
L4virtio::Svr::Block_request< Ds_data >	1619
L4virtio::Svr::Data_buffer	1621
L4virtio::Svr::Dev_config	1626
L4virtio::Svr::Dev_features	1637
L4virtio::Svr::Dev_status	1640
L4virtio::Svr::Device_t< DATA >	1646
L4virtio::Svr::Driver_mem_list_t< DATA >	1652
L4virtio::Svr::Driver_mem_region_t< DATA >	1659
L4virtio::Svr::Request_processor	1666
L4virtio::Svr::Virtqueue::Head_desc	1679
L4virtio::Virtqueue	1681
L4virtio::Driver::Virtqueue	1598
L4virtio::Svr::Virtqueue	1672
L4virtio::Virtqueue::Avail	1694
L4virtio::Virtqueue::Avail::Flags	1695
L4virtio::Virtqueue::Desc	1697
L4virtio::Virtqueue::Desc::Flags	1699
L4virtio::Virtqueue::Used	1703
L4virtio::Virtqueue::Used::Flags	1704
L4virtio::Virtqueue::Used_elem	1706
l4virtio_block_config_t	1707
l4virtio_block_discard_t	1709
l4virtio_block_header_t	1710
l4virtio_config_hdr_t	1711
l4virtio_config_queue_t	1713
L4vbus::Pm< Device >	1568
L4vbus::Device	1531
L4vbus::Gpio_module	1540
L4vbus::Gpio_pin	1547
L4vbus::Icu	1555
L4vbus::Pci_dev	1557
L4vbus::Pci_host_bridge	1562
L4virtio::Ptr< void >	1604
cxx::Ref_ptr< L4Re::Vfs::File >	830
cxx::Ref_ptr< Mount_tree >	830
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_in, CLASS >	1004
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_out, CLASS >	1004
L4::lpc::Msg::Svr_val_ops< L4::lpc::Snd_fpage, Dir_in, CLASS >	1004
L4::lpc::Msg::Svr_val_ops< typename _Elem< A * > ::svr_type, typename Direction< A * > ::type, typename Class< typename Detail::_Plain< A * > ::type > ::type >	1004
L4::lpc::Msg::Svr_val_ops< typename _Elem< A > ::svr_type, typename Direction< A > ::type, typename Class< typename Detail::_Plain< A > ::type > ::type >	1004
L4::lpc::Msg::Svr_val_ops< typename _Elem< A const * > ::svr_type, typename Direction< A const * > ::type, typename Class< typename Detail::_Plain< A const * > ::type > ::type >	1004
L4::lpc::Msg::Svr_val_ops< typename _Elem< Array_ref< A, LEN > & > ::svr_type, typename Direction< Array_ref< A, LEN > & > ::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > & > ::type > ::type >	1004
L4::lpc::Msg::Svr_val_ops< typename _Elem< Array_ref< A, LEN > > ::svr_type, typename Direction< Array_ref< A, LEN > > ::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > > ::type > ::type >	1004
L4::lpc::Msg::Svr_val_ops< typename _Elem< T > ::svr_type, typename Direction< T > ::type, typename Class< typename Detail::_Plain< T > ::type > ::type >	1004
cxx::Bitmap_base::Word< Bits >	753
cxx::Bitmap_base::Word< Size >	753

Chapter 10

Data Structure Index

10.1 Data Structures

Here are the data structures with brief descriptions:

cxx::Auto_ptr< T >	
Smart pointer with automatic deletion	695
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	
AVL tree based associative container	700
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	
AVL set for simple compareable items	704
cxx::Avl_tree< Node, Get_key, Compare >	
A generic AVL tree	707
cxx::Avl_tree_node	
Node of an AVL tree	712
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	
Basic slab allocator	715
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	718
cxx::Bitfield< T, LSB, MSB >	
Definition for a member (part) of a bit field	723
cxx::Bitfield< T, LSB, MSB >::Value< TT >	
Internal helper type	733
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	
Internal helper type	734
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	
Internal helper type	736
cxx::Bitmap< BITS >	
A static bit map	737
cxx::Bitmap_base	
Basic bitmap abstraction	741
cxx::Bitmap_base::Bit	
A writeable bit in a bitmap	752
cxx::Bitmap_base::Char< BITS >	
Helper abstraction for a byte contained in the bitmap	753
cxx::Bitmap_base::Word< BITS >	
Helper abstraction for a word contained in the bitmap	753
cxx::Bits::Avl_map_get_key< KEY_TYPE >	
Key-getter for Avl_map	755
cxx::Bits::Avl_set_get_key< KEY_TYPE >	
Internal, key-getter for Avl_set nodes	755

cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	
Internal: AVL set with internally managed nodes	756
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	
A smart pointer to a tree item	767
cxx::Bits::Basic_list< POLICY >	
Internal: Common functions for all head-based list implementations	769
cxx::Bits::Bst< Node, Get_key, Compare >	
Basic binary search tree (BST)	771
cxx::Bits::Bst_node	
Basic type of a node in a binary search tree (BST)	785
cxx::Bits::Direction	
The direction to go in a binary search tree	787
cxx::Bits::Smart_ptr_list< ITEM >	
List of smart-pointer-managed objects	789
cxx::Bits::Smart_ptr_list_item< T, STORE_T >	
List item for an arbitrary item in a Smart_ptr_list	791
cxx::H_list< T, POLICY >	
General double-linked list of unspecified cxx::H_list_item elements	792
cxx::H_list_item_t< ELEM_TYPE >	
Basic element type for a double-linked H_list	799
cxx::H_list_t< T >	
Double-linked list of typed H_list_item_t elements	801
cxx::List< D, Alloc >	
Doubly linked list, with internal allocation	804
cxx::List< D, Alloc >::Iter	
Iterator	807
cxx::List_alloc	
Standard list-based allocator	808
cxx::List_item	
Basic list item	812
cxx::List_item::Iter	
Iterator for a list of ListItem -s	818
cxx::List_item::T_iter< T, Poly >	
Iterator for derived classes from ListItem	821
cxx::Lt_func< Obj >	
Generic comparator class that defaults to the less-than operator	824
cxx::New_allocator< _Type >	
Standard allocator based on <code>operator new ()</code>	824
cxx::Nothrow	
Helper type to distinguish the <code>operator new</code> version that does not throw exceptions	825
cxx::Pair< First, Second >	
Pair of two values	826
cxx::Pair_first_compare< Cmp, Typ >	
Comparison functor for Pair	827
cxx::Ref_obj_list_item< T >	
Item for list linked via cxx::Ref_ptr with default reference counting	829
cxx::Ref_ptr< T, CNT >	
A reference-counting pointer with automatic cleanup	830
cxx::S_list< T, POLICY >	
Simple single-linked list	836
cxx::Slab< Type, Slab_size, Max_free, Alloc >	
Slab allocator for object of type <i>Type</i>	839
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	843
cxx::static_vector< T, IDX >	
Simple encapsulation for a dynamically allocated array	846
cxx::String	
Allocation free string class with explicit length field	848

cxx::Weak_ref< T >	855
Typed weak reference to an object of type <code>T</code>	
cxx::Weak_ref_base	857
Generic (base) weak reference to some object	
Elf32_Dyn	860
ELF32 dynamic entry	
Elf32_Ehdr	861
ELF32 header	
Elf32_Phdr	863
ELF32 program header	
Elf32_Shdr	864
ELF32 section header - figure 1-9, page 1-9	
Elf32_Sym	865
ELF32 symbol table entry	
Elf64_Dyn	866
ELF64 dynamic entry	
Elf64_Ehdr	867
ELF64 header	
Elf64_Phdr	869
ELF64 program header	
Elf64_Shdr	870
ELF64 section header	
Elf64_Sym	871
ELF64 symbol table entry	
L4::Alloc_list	872
A simple list-based allocator	
L4::Arm_smccc	873
Wrapper for function calls that follow the ARM SMC/HVC calling convention	
L4::Base_exception	874
Base class for all exceptions, thrown by the L4Re framework	
L4::Basic_registry	876
This registry returns the corresponding server object based on the label of an lpc_gate	
L4::Bounds_error	878
Access out of bounds	
L4::Cap< T >	881
C++ interface for capabilities	
L4::Cap_base	886
Base class for all kinds of capabilities	
L4::Com_error	897
Error conditions during IPC	
L4::Debugger	899
C++ kernel debugger API	
L4::Element_already_exists	906
Exception for duplicate element insertions	
L4::Element_not_found	908
Exception for a failed lookup (element not found)	
L4::Exception	910
Exception interface	
L4::Exception_tracer	911
Back-trace support for exceptions	
L4::Factory	913
C++ Factory interface to create kernel objects	
L4::Factory::Lstr	924
Special type to add a pascal string into the factory create stream	
L4::Factory::Nil	925
Special type to add a void argument into the factory create stream	
L4::Factory::S	926
Stream class for the create() argument stream	

L4::lcu	
C++ lcu interface	931
L4::lcu::Info	
This class encapsulates information about an ICU	939
L4::Invalid_capability	
Indicates that an invalid object was invoked	940
L4::lo_pager	
lo_pager interface	944
L4::lommu	
Interface for IO-MMUs used for DMA remapping	946
L4::IOModifier	
Modifier class for the IO stream	949
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	
Array data type for dynamically sized arrays in RPCs	950
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	
Server-side copy in buffer for Array	953
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	
Array reference data type for arrays located in the message	954
L4::lpc::As_value< T >	
Pass the argument as plain data value	956
L4::lpc::Buf_item	
RPC warpper for a receive item	957
L4::lpc::Call	
RPC attribute for a standard RPC call	958
L4::lpc::Call_t< RIGHTS >	
RPC attribute for an RPC call with required rights	959
L4::lpc::Call_zero_send_timeout	
RPC attribute for an RPC call, with zero send timeout	961
L4::lpc::Cap< T >	
Capability type for RPC interfaces (see L4::Cap<T>)	962
L4::lpc::Gen_fpage< T >	
Generic RPC wrapper for L4 flex-pages	965
L4::lpc::In_out< T >	
Mark an argument as in-out argument	969
L4::lpc::Iostream	
Input/Output stream for IPC [un]marshalling	969
L4::lpc::Istream	
Input stream for IPC unmarshalling	976
L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >	
Defines client-side handling of 'MTYPE' as RPC argument	986
L4::lpc::Msg::Cls_buffer	
Marker type for receive buffer values	987
L4::lpc::Msg::Cls_data	
Marker type for data values	989
L4::lpc::Msg::Cls_item	
Marker type for item values	990
L4::lpc::Msg::Dir_in	
Marker type for input values	991
L4::lpc::Msg::Dir_out	
Marker type for output values	992
L4::lpc::Msg::Do_in_data	
Marker for Input data	993
L4::lpc::Msg::Do_in_items	
Marker for Input items	994
L4::lpc::Msg::Do_out_data	
Marker for Output data	995
L4::lpc::Msg::Do_out_items	
Marker for Output items	996

L4::lpc::Msg::Do_rcv_buffers	
Marker for receive buffers	997
L4::lpc::Msg::Elem< Array< A, LEN > &>	
Array as output argument	999
L4::lpc::Msg::Elem< Array< A, LEN > >	
Array as input arguments	1000
L4::lpc::Msg::Elem< Array_ref< A, LEN > &>	
Array_ref as output argument	1001
L4::lpc::Msg::Is_valid_rpc_type< T >	
Type trait defining a valid RPC parameter type	1002
L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	
Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function	1004
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	
Defines server-side handling for MTYPE server arguments	1004
L4::lpc::Msg_ptr< T >	
Pointer to an element of type T in an lpc::lstream	1005
L4::lpc::Opt< T >	
Attribute for defining an optional RPC argument	1006
L4::lpc::Ostream	
Output stream for IPC marshalling	1008
L4::lpc::Out< T >	
Mark an argument as a output value in an RPC signature	1014
L4::lpc::Ret_array< T >	
Dynamically sized output array of type T	1015
L4::lpc::Send_only	
RPC attribute for a send-only RPC	1016
L4::lpc::Small_buf	
A receive item for receiving a single capability	1016
L4::lpc::Snd_item	
RPC wrapper for a send item	1018
L4::lpc::Str_cp_in< T >	
Abstraction for extracting a zero-terminated string from an lpc::lstream	1018
L4::lpc::Varg	
Variably sized RPC argument	1020
L4::lpc::Varg_list< MAX >	
Self-contained list of variable-sized RPC parameters	1025
L4::lpc::Varg_list_ref	
List of variable-sized RPC parameters as received by the server	1026
L4::lpc_gate	
The C++ IPC gate interface	1028
L4::lpc_svr::Br_manager_no_buffers	
Empty implementation of Server_iface	1031
L4::lpc_svr::Compound_reply	
Mix in for LOOP_HOOKS to always use compound reply and wait	1035
L4::lpc_svr::Default_loop_hooks	
Default LOOP_HOOKS	1036
L4::lpc_svr::Default_setup_wait	
Mix in for LOOP_HOOKS for setup_wait no op	1037
L4::lpc_svr::Default_timeout	
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout	1038
L4::lpc_svr::Direct_dispatch< R >	
Direct dispatch helper, for forwarding dispatch calls a registry <i>R</i>	1039
L4::lpc_svr::Direct_dispatch< R * >	
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry <i>R</i>	1041
L4::lpc_svr::Exc_dispatch< R, Exc >	
Dispatch helper wrapping try {} catch {} around the dispatch call	1042

L4::lpc_svr::Ignore_errors	
Mixin for LOOP_HOOKS to ignore IPC errors	1043
L4::lpc_svr::Server_iface	
Interface for server-loop related functions	1044
L4::lpc_svr::Timed_work< HOOKS >	
DEPRECATED	1053
L4::lpc_svr::Timeout	
Callback interface for Timeout_queue	1053
L4::lpc_svr::Timeout_queue	
Timeout queue to be used in l4re server loop	1056
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	
Loop hooks mixin for integrating a timeout queue into the server loop	1060
L4::lrq	
C++ lrq interface	1063
L4::lrq_eoi	
Interface for sending an acknowledge message to an object	1070
L4::lrq_handler_object	
Server object base class for handling IRQ messages	1073
L4::lrq_mux	
IRQ multiplexer for shared IRQs	1076
L4::Kip::Mem_desc	
Memory descriptors stored in the kernel interface page	1079
L4::Kobject	
Base class for all kinds of kernel objects and remote objects, referenced by capabilities	1088
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >	
Helper class to create an L4Re interface class that is derived from two base classes (see L4::Kobject_t)	1092
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >	
Helper class to create an L4Re interface class that is derived from three base classes (see L4::Kobject_t)	1096
L4::Kobject_demand< T >	
Get the combined server-side resource requirements for all type T..	1099
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	
Helper class to create an L4Re interface class that is derived from a single base class	1100
L4::Kobject_typeid< T >	
Meta object for handling access to type information of Kobjects	1102
L4::Kobject_typeid< void >	
Minimalistic ID for void interface	1105
L4::Kobject_x< Derived, ARGS >	
Generic Kobject inheritance template	1106
L4::Meta	
Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects	1107
L4::Out_of_memory	
Exception signalling insufficient memory	1111
L4::Pager	
Pager interface including the lo_pager interface	1114
L4::Platform_control	
L4 C++ interface for controlling platform-wide properties	1118
L4::Poll_timeout_kipclock	
A polling timeout based on the L4Re clock	1122
L4::Proto_t< P >	
Data type for defining protocol numbers	1125
L4::Rcv_endpoint	
Interface for kernel objects that allow to receive IPC from them	1127
L4::Registry_iface	
Abstract interface for object registries	1131

L4::Runtime_error	
Exception for an abstract runtime error	1136
L4::Scheduler	
C++ interface of the Scheduler kernel object	1139
L4::Semaphore	
Kernel-provided semaphore object	1144
L4::Server< LOOP_HOOKS >	
Basic server loop for handling client requests	1149
L4::Server_object	
Abstract server object to be used with L4::Server and L4::Basic_registry	1152
L4::Server_object_t< IFACE, BASE >	
Base class (template) for server implementing server objects	1154
L4::Server_object_x< Derived, IFACE, BASE >	
Helper class to implement p_dispatch based server objects	1158
L4::Smart_cap< T, SMART >	
Smart capability class	1160
L4::String	
A null-terminated string container class	1164
L4::Task	
C++ interface of the Task kernel object	1164
L4::Thread	
C++ L4 kernel thread interface	1173
L4::Thread::Attr	
Thread attributes used for control_commit()	1183
L4::Thread::Modify_senders	
Wrapper class for modifying senders	1186
L4::Triggerable	
Interface that allows an object to be triggered by some source	1188
L4::Type_info	
Dynamic Type Information for L4Re Interfaces	1192
L4::Type_info::Demand	
Data type for expressing the needed receive buffers at the server-side of an interface	1193
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	
Template type statically describing demand of receive buffers	1196
L4::Type_info::Demand_union_t< D1, D2 >	
Template type statically describing the combination of two Demand object	1198
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	
Empty list of RPCs	1201
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >	
Find the given RPC in the list	1202
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	
Non-empty list of RPCs	1203
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	
Find the given RPC in the list	1204
L4::Typeid::Detail::Rpc_end	
Internal end-of-list marker	1205
L4::Typeid::P_dispatch< LIST >	
Use for protocol based dispatch stage	1206
L4::Typeid::Raw_ipc< CLASS >	
RPCs list for passing raw incoming IPC to the server object	1207
L4::Typeid::Rpc_nocode< OPERATION >	
List of RPCs of an interface using a single operation without an opcode	1207
L4::Typeid::Rpc< RPCS >	
Standard list of RPCs of an interface	1210
L4::Typeid::Rpc_code< OPCODE_TYPE >	
List of RPCs of an interface using a special opcode type	1212
L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >	1213

L4::Typeid::Rpcsys< ARG >	List of RPCs typically used for kernel interfaces	1215
L4::Types::Bool< V >	Boolean meta type	1217
L4::Types::False	False meta value	1218
L4::Types::Flags< BITS_ENUM, UNDERLYING >	Template for defining typical Flags bitmaps	1220
L4::Types::Same< A, B >	Compare two data types for equality	1225
L4::Types::True	True meta value	1227
L4::Unknown_error	Exception for an unknown condition	1229
L4::Vcon	C++ L4 Vcon interface	1232
L4::Vm	Virtual machine	1239
l4_buf_regs_t	Encapsulation of the buffer-registers block in the UTCB	1242
l4_exc_regs_t	UTCB structure for exceptions	1243
l4_fpage_t	L4 flexpage type	1246
l4_icu_info_t	Info structure for an ICU	1246
l4_icu_msi_info_t	Info to use for a specific MSI	1248
l4_kernel_info_mem_desc_t	Memory descriptor data structure	1249
l4_kernel_info_t	L4 Kernel Interface Page	1250
l4_msg_regs_t	Encapsulation of the message-register block in the UTCB	1252
l4_msgtag_t	Message tag data structure	1253
l4_sched_cpu_set_t	CPU sets	1255
l4_sched_param_t	Scheduler parameter set	1258
l4_snd_fpage_t	Send-flex-page types	1259
l4_thread_regs_t	Encapsulation of the thread-control-register block of the UTCB	1260
l4_timeout_s	Basic timeout specification	1261
l4_timeout_t	Timeout pair	1262
l4_tracebuffer_status_t	Trace-buffer status	1263
l4_tracebuffer_status_window_t	Trace-buffer status window descriptor	1265
l4_vcon_attr_t	Vcon attribute structure	1266
l4_vcpu_ipc_regs_t	VCPU message registers	1267
l4_vcpu_regs_t	VCPU registers	1268

l4_vcpu_state_t	State of a vCPU	1272
l4_vhw_descriptor	Virtual hardware devices description	1274
l4_vhw_entry	Description of a device	1277
l4_vm_svm_vmcb_control_area	VMCB structure for SVM VMs	1280
l4_vm_svm_vmcb_state_save_area	State save area structure for SVM VMs	1280
l4_vm_svm_vmcb_state_save_area_seg	State save area segment selector struct	1282
l4_vm_svm_vmcb_t	Control structure for SVM VMs	1282
l4_vm_tz_state	State structure for TrustZone VMs	1284
L4Re::Cap_alloc	Capability allocator interface	1284
L4Re::Console	Console class	1289
L4Re::Dataspace	Interface for memory-like objects	1291
L4Re::Dataspace::Stats	Information about the dataspace	1300
L4Re::Debug_obj	Debug interface	1301
L4Re::Dma_space	DMA Address Space	1303
L4Re::Env	C++ interface of the initial environment that is provided to an L4 task	1308
L4Re::Event	Event class	1321
L4Re::Event_buffer_t< PAYLOAD >	Event buffer class	1325
L4Re::Event_buffer_t< PAYLOAD >::Event	Event structure used in buffer	1328
L4Re::Inhibitor	Set of inhibitor locks, which inhibit specific actions when held	1329
L4Re::Log	Log interface class	1334
L4Re::Mem_alloc	Memory allocation interface	1337
L4Re::Mmio_space	Interface for memory-like address space accessible via IPC	1342
L4Re::Namespace	Name-space interface	1347
L4Re::Ned::Cmd_control	Direct control interface for Ned	1353
L4Re::Parent	Parent interface	1355
L4Re::Rm	Region map	1357
L4Re::Smart_cap_auto< Unmap_flags >	Helper for Auto_cap and Auto_del_cap	1370
L4Re::Smart_count_cap< Unmap_flags >	Helper for Ref_cap and Ref_del_cap	1371
L4Re::Util::Auto_cap< T >	Automatic capability that implements automatic free and unmap of the capability selector	1372

L4Re::Util::Auto_del_cap< T >	
Automatic capability that implements automatic free and unmap+delete of the capability selector	1373
L4Re::Util::Br_manager	
Buffer-register (BR) manager for L4::Server	1374
L4Re::Util::Br_manager_hooks	
Predefined server-loop hooks for a server loop using the Br_manager	1379
L4Re::Util::Br_manager_timeout_hooks	
Predefined server-loop hooks for a server with using the Br_manager and a timeout queue	1380
L4Re::Util::Cap_alloc_base	
Capability allocator	1383
L4Re::Util::Counter< COUNTER >	
Counter for Counting_cap_alloc with variable data width	1384
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	
Internal reference-counting cap allocator	1384
L4Re::Util::Dataspace_svr	
Dataspace server class	1391
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	
An event buffer consumer	1401
L4Re::Util::Event_buffer_t< PAYLOAD >	
Event_buffer utility class	1405
L4Re::Util::Event_svr< SVR >	
Convenience wrapper for implementing an event server	1409
L4Re::Util::Event_t< PAYLOAD >	
Convenience wrapper for getting access to an event object	1411
L4Re::Util::Item_alloc_base	
Item allocator	1415
L4Re::Util::Names::Name	
Name class	1416
L4Re::Util::Names::Name_space	
Abstract server-side implementation of the L4::Namespace interface	1417
L4Re::Util::Object_registry	
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread	1421
L4Re::Util::Ref_cap< T >	
Automatic capability that implements automatic free and unmap of the capability selector	1427
L4Re::Util::Ref_del_cap< T >	
Automatic capability that implements automatic free and unmap+delete of the capability selector	1428
L4Re::Util::Registry_server< LOOP_HOOKS >	
A server loop object which has a Object_registry included	1429
L4Re::Util::Smart_cap_auto< Unmap_flags >	
Helper for Auto_cap and Auto_del_cap	1433
L4Re::Util::Smart_count_cap< Unmap_flags >	
Helper for Ref_cap and Ref_del_cap	1434
L4Re::Util::Vcon_svr< SVR >	
Console server template class	1435
L4Re::Util::Video::Goos_svr	
Goos server class	1436
L4Re::Vfs::Be_file	
Boiler plate class for implementing an open file for L4Re::Vfs	1440
L4Re::Vfs::Be_file_system	
Boilerplate class for implementing a L4Re::Vfs::File_system	1444
L4Re::Vfs::Directory	
Interface for a POSIX file that is a directory	1447
L4Re::Vfs::File	
The basic interface for an open POSIX file	1452
L4Re::Vfs::File_system	
Basic interface for an L4Re::Vfs file system	1454

L4Re::Vfs::Fs	POSIX File-system related functionality	1456
L4Re::Vfs::Generic_file	The common interface for an open POSIX file	1461
L4Re::Vfs::Mman	Interface for the POSIX memory management	1466
L4Re::Vfs::Ops	Interface for the POSIX backends for an application	1467
L4Re::Vfs::Regular_file	Interface for a POSIX file that provides regular file semantics	1470
L4Re::Vfs::Special_file	Interface for a POSIX file that provides special file semantics	1475
L4Re::Video::Color_component	A color component	1478
L4Re::Video::Goos	A goos	1482
L4Re::Video::Goos::Info	Information structure of a goos	1488
L4Re::Video::Pixel_info	Pixel information	1490
L4Re::Video::View	View	1497
L4Re::Video::View::Info	Information structure of a view	1502
l4re_aux_t	Auxiliary descriptor	1505
l4re_ds_stats_t	Information about the data space	1506
l4re_elf_aux_mword_t	Auxiliary vector element for a single unsigned data word	1506
l4re_elf_aux_t	Generic header for each auxiliary vector element	1507
l4re_elf_aux_vma_t	Auxiliary vector element for a reserved virtual memory area	1508
l4re_env_cap_entry_t	Entry in the L4Re environment array for the named initial objects	1509
l4re_env_t	Initial environment data structure	1511
l4re_event_t	Event structure used in buffer	1512
l4re_video_color_component_t	Color component structure	1513
l4re_video_goos_info_t	Goos information structure	1514
l4re_video_pixel_info_t	Pixel_info structure	1515
l4re_video_view_info_t	View information structure	1516
l4re_video_view_t	C representation of a goos view	1518
l4util_idt_desc_t	IDT entry	1519
l4util_idt_header_t	Header of an IDT table	1520
l4util_mb_addr_range_t	INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached	1521

l4util_mb_apm_t	
APM BIOS info	1522
l4util_mb_drive_t	
Drive Info structure	1522
l4util_mb_info_t	1524
l4util_mb_mod_t	1526
l4util_mb_vbe_ctrl_t	
VBE controller information	1527
l4util_mb_vbe_mode_t	
VBE mode information	1528
L4vbus::Device	
Device on a L4vbus::Vbus	1531
L4vbus::Gpio_module	
A Gpio_module groups multiple GPIO pins together	1540
L4vbus::Gpio_module::Pin_slice	
A slice of the pins provided by this module	1546
L4vbus::Gpio_pin	
A GPIO pin	1547
L4vbus::lcu	
Vbus Interrupt controller API	1555
L4vbus::Pci_dev	
A PCI device	1557
L4vbus::Pci_host_bridge	
A Pci host bridge	1562
L4vbus::Pm< DEC >	
Power-management API mixin	1568
L4vbus::Vbus	
The virtual bus (Vbus) interface	1569
l4vbus_device_t	
Detailed information about a vbus device	1576
l4vbus_resource_t	
Description of a single vbus resource	1577
L4vcpu::State	
C++ implementation of state word in the vCPU area	1578
L4vcpu::Vcpu	
C++ implementation of the vCPU save state area	1580
L4virtio::Device	
IPC interface for virtio over L4 IPC	1593
L4virtio::Driver::Virtqueue	
Driver-side implementation of a Virtqueue	1598
L4virtio::Ptr< T >	
Pointer used in virtio descriptors	1604
L4virtio::Svr::Bad_descriptor	
Exception used by Queue to indicate descriptor errors	1608
L4virtio::Svr::Block_dev< Ds_data >	
Base class for virtio block devices	1610
L4virtio::Svr::Block_request< Ds_data >	
A request to read or write data	1619
L4virtio::Svr::Data_buffer	
Abstract data buffer	1621
L4virtio::Svr::Dev_config	
Abstraction for L4-Virtio device config memory	1626
L4virtio::Svr::Dev_features	
Type for device feature bitmap	1637
L4virtio::Svr::Dev_status	
Type of the device status register	1640
L4virtio::Svr::Device_t< DATA >	
Server-side L4-VIRTIO device stub	1646

L4virtio::Svr::Driver_mem_list_t< DATA >	
List of driver memory regions assigned to a single L4-VIRTIO transport instance	1652
L4virtio::Svr::Driver_mem_region_t< DATA >	
Region of driver memory, that shall be managed locally	1659
L4virtio::Svr::Request_processor	
Encapsulate the state for processing a VIRTIO request	1666
L4virtio::Svr::Virtqueue	
Virtqueue implementation for the device	1672
L4virtio::Svr::Virtqueue::Head_desc	
VIRTIO request, essentially a descriptor from the available ring	1679
L4virtio::Virtqueue	
Low-level Virtqueue	1681
L4virtio::Virtqueue::Avail	
Type of available ring, this is read-only for the host	1694
L4virtio::Virtqueue::Avail::Flags	
Flags of the available ring	1695
L4virtio::Virtqueue::Desc	
Descriptor in the descriptor table	1697
L4virtio::Virtqueue::Desc::Flags	
Type for descriptor flags	1699
L4virtio::Virtqueue::Used	
Used ring	1703
L4virtio::Virtqueue::Used::Flags	
Flags for the used ring	1704
L4virtio::Virtqueue::Used_elem	
Type of an element of the used ring	1706
l4virtio_block_config_t	
Device configuration for block devices	1707
l4virtio_block_discard_t	
Structure used for the write zeroes and discard commands	1709
l4virtio_block_header_t	
Header structure of a request for a block device	1710
l4virtio_config_hdr_t	
L4-VIRTIO config header, provided in shared data space	1711
l4virtio_config_queue_t	
Queue configuration entry	1713

Chapter 11

File Index

11.1 File List

Here is a list of all documented files with brief descriptions:

pkg/l4re-core/ned/lib/include/ cmd_control	??
pkg/l4re-core/ned/lib/include/ Makefile	??
amd64/l4/sys/ __kip-arch.h	??
amd64/l4/sys/ __vcpu-arch.h	??
amd64/l4/sys/ cache.h	
Cache functions	2066
amd64/l4/sys/ consts.h	
Common L4 constants, amd64 version	2080
amd64/l4/sys/ ktrace_events.h	??
amd64/l4/sys/ l4int.h	
Fixed sized integer types, amd64 version	2173
amd64/l4/sys/ linkage.h	
Linkage	1786
amd64/l4/sys/ segment.h	
Segment handling	1761
amd64/l4/sys/ utcb.h	
UTCB definitions for amd64	2228
amd64/l4/sys/ vm.h	??
amd64/l4/util/ apic.h	
APIC for AMD64	1715
amd64/l4/util/ bitops_arch.h	
Amd64 bit manipulation functions	1792
amd64/l4/util/ cpu.h	
CPU related functions	1800
amd64/l4/util/ idt.h	
IDT related functions	1726
amd64/l4/util/ irq.h	
Some PIC and hardware interrupt related functions	1896
amd64/l4/util/ l4_macros.h	
Main function	1805
amd64/l4/util/ mbi_argv.h	
Command line handling	1808
amd64/l4/util/ perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1729
amd64/l4/util/ port_io.h	
Port I/O functions	1771

amd64/l4/util/ rdtsc.h	
Time stamp counter related functions	1741
amd64/l4/util/ spin.h	
Spinning for amd64	1751
amd64/l4/util/ stack_impl.h	
Stack utilities for amd64	1812
amd64/l4/util/ util.h	
Utilities, amd64 version	1753
amd64/l4f/l4/sys/ ipc.h	??
amd64/l4f/l4/sys/ segment.h	
L4f specific fs/gs manipulation	1758
amd64/l4f/l4/util/ port_io.h	
Port I/O functions	1770
amd64/l4f/l4/util/ setjmp.h	
Inter-thread setjmp/longjmp	1779
arm/l4/sys/ __kip-arch.h	??
arm/l4/sys/ __vcpu-arch.h	??
arm/l4/sys/ atomic.h	
Atomic memory modifications	2261
arm/l4/sys/ cache.h	
Cache functions	2063
arm/l4/sys/ consts.h	
Common L4 constants, arm version	2079
arm/l4/sys/ ktrace_events.h	??
arm/l4/sys/ l4int.h	
Fixed sized integer types, arm version	2173
arm/l4/sys/ linkage.h	
Linkage	1785
arm/l4/sys/ mem_op.h	
Memory access functions (ARM specific)	1787
arm/l4/sys/ thread.h	
ARM-specific thread related definitions	2339
arm/l4/sys/ utcb.h	
UTCB definitions for ARM	2226
arm/l4/sys/ vm.h	
ARM virtualization interface	1790
arm/l4/util/ bitops_arch.h	
ARM specific implementation of bitops functions	1791
arm/l4/util/ cpu.h	
CPU related functions	1799
arm/l4/util/ irq.h	
ARM specific implementation of irq functions	1895
arm/l4/util/ l4_macros.h	
Main function	1804
arm/l4/util/ mbi_argv.h	
Multiboot	1807
arm/l4/util/ stack_impl.h	
Stack utilities, arm version	1811
arm/l4f/l4/sys/ ipc.h	
L4 IPC System Calls, ARM	2151
arm/l4f/l4/sys/ syscall_defs.h	
Syscall entry definitions	1814
contrib/libio-io/l4/io/ io.h	1815
contrib/libio-io/l4/io/ types.h	??
l4/cxx/ alloc.h	
Alloc list	2248
l4/cxx/ arith	??

l4/cxx/ atomic.h	
Atomic template	2262
l4/cxx/ auto_ptr	??
l4/cxx/ avl_map	
AVL map	1825
l4/cxx/ avl_set	
AVL set	1828
l4/cxx/ avl_tree	
AVL tree	1832
l4/cxx/ basic_ostream	
Basic IO stream	1838
l4/cxx/ basic_vector.h	
Basic vector	1842
l4/cxx/ bitfield	??
l4/cxx/ bitmap	??
l4/cxx/ dlist	??
l4/cxx/ exceptions	
Base exceptions	1853
l4/cxx/ hlist	??
l4/cxx/ iostream	
IO Stream	1858
l4/cxx/ ipc_helper	
IPC helper	1859
l4/cxx/ ipc_server	
IPC server loop	2098
l4/cxx/ ipc_stream	
IPC stream	1861
l4/cxx/ ipc_timeout_queue	??
l4/cxx/ l4iostream	
L4 IO stream	1882
l4/cxx/ l4types.h	
L4 Types	1883
l4/cxx/ list	??
l4/cxx/ list_alloc	??
l4/cxx/ main_thread	
Main thread	1885
l4/cxx/ minmax	??
l4/cxx/ observer	??
l4/cxx/ pair	
Pair implementation	1886
l4/cxx/ ref_ptr	??
l4/cxx/ ref_ptr_list	??
l4/cxx/ slab_alloc	??
l4/cxx/ slist	??
l4/cxx/ static_container	??
l4/cxx/ static_vector	??
l4/cxx/ std_alloc	??
l4/cxx/ std_exc_io	
Base exceptions std stream operator	1888
l4/cxx/ std_ops	??
l4/cxx/ string	??
l4/cxx/ string.h	
String	1890
l4/cxx/ thread	
Thread implementation	2217
l4/cxx/ type_list	??
l4/cxx/ type_traits	??
l4/cxx/ unique_ptr	??

l4/cxx/unique_ptr_list	??
l4/cxx/utls	??
l4/cxx/weak_ref	??
l4/cxx/bits/bst.h	
AVL tree	1843
l4/cxx/bits/bst_base.h	
AVL tree	1847
l4/cxx/bits/bst_iter.h	
AVL tree	1850
l4/cxx/bits/list_basics.h	??
l4/cxx/bits/smart_ptr_list.h	??
l4/cxx/bits/type_traits.h	??
l4/irq/irq.h	
IRQ handling routines	1892
l4/l4re_vfs/backend	??
l4/l4re_vfs/vfs.h	??
l4/l4re_vfs/impl/default_ops_impl.h	??
l4/l4re_vfs/impl/fd_store.h	??
l4/l4re_vfs/impl/fd_store_impl.h	??
l4/l4re_vfs/impl/ns_fs.h	??
l4/l4re_vfs/impl/ns_fs_impl.h	??
l4/l4re_vfs/impl/ro_file.h	??
l4/l4re_vfs/impl/ro_file_impl.h	??
l4/l4re_vfs/impl/vcon_stream.h	??
l4/l4re_vfs/impl/vcon_stream_impl.h	??
l4/l4re_vfs/impl/vfs_impl.h	??
l4/l4virtio/l4virtio	??
l4/l4virtio/virtio.h	??
l4/l4virtio/virtio_block.h	??
l4/l4virtio/virtqueue	??
l4/l4virtio/server/l4virtio	??
l4/l4virtio/server/virtio	??
l4/l4virtio/server/virtio-block	??
l4/libedid/edid.h	1906
l4/re/cap_alloc	
Abstract capability-allocator interface	1944
l4/re/console	??
l4/re/consts	
Constants	1951
l4/re/consts.h	
Constants	2082
l4/re/dataspace	
Dataspace interface	1953
l4/re/dataspace-sys.h	
Dataspace protocol definition	1955
l4/re/debug	
Debug interface	1956
l4/re/dma_space	1958
l4/re/elf_aux.h	
Auxiliary information for binaries	1961
l4/re/env	
Environment interface	1963
l4/re/env.h	
Environment interface	1965
l4/re/error_helper	
Error helper	1969
l4/re/event	??
l4/re/event-sys.h	??

l4/re/event.h	
Events	1917
l4/re/event_enums.h	??
l4/re/inhibitor	??
l4/re/inhibitor-sys.h	??
l4/re/l4aux.h	
Auxiliary definitions	1984
l4/re/log	
Log interface	1985
l4/re/log-sys.h	
Log protocol definition	1987
l4/re/mem_alloc	
Memory allocator interface	1988
l4/re/mem_alloc-sys.h	
Memory allocator protocol definitions	1990
l4/re/mmio_space	??
l4/re/namespace	
Namespace interface	1992
l4/re/namespace-sys.h	
Namespace protocol definitions	1994
l4/re/parent	
Parent interface	1995
l4/re/parent-sys.h	
Parent protocol definition	1997
l4/re/protocols.h	
L4Re Protocol Constants (C version)	1998
l4/re/rm	
Region mapper interface	2000
l4/re/rm-sys.h	
Region mapper protocol definitions	2005
l4/re/shared_cap	
Shared_cap / Shared_del_cap	2006
l4/re/unique_cap	
Unique_cap / Unique_del_cap	2011
l4/re/c/dataspace.h	
Data space C interface	1908
l4/re/c/debug.h	
Debug C interface	1910
l4/re/c/dma_space.h	
DMA space C interface	1912
l4/re/c/event.h	
Event C interface	1915
l4/re/c/event_buffer.h	??
l4/re/c/inhibitor.h	??
l4/re/c/log.h	
Log C interface	1919
l4/re/c/mem_alloc.h	
Memory allocator C interface	1921
l4/re/c/namespace.h	
Namespace functions, C interface	1924
l4/re/c/rm.h	
Region map interface, C interface	1926
l4/re/c/util/cap_alloc.h	
Capability allocator C interface	1930
l4/re/c/util/kumem_alloc.h	
Kumem allocator utility C interface	1931
l4/re/c/util/video/goos_fb.h	
Framebuffer utility functionality	1933

l4/re/c/video/colors.h	1935
l4/re/c/video/goos.h	1937
l4/re/c/video/view.h	1940
l4/re/impl/dataspace_impl.h	
Dataspace client stub implementation	1975
l4/re/impl/mem_alloc_impl.h	
Memory allocator client stub implementation	1977
l4/re/impl/namespace_impl.h	
Namespace client stub implementation	1979
l4/re/impl/rm_impl.h	
Region map client stub implementation	1981
l4/re/util/bitmap_cap_alloc	
Bitmap capability allocator	2016
l4/re/util/br_manager	??
l4/re/util/cap	
Capability utility functions	2018
l4/re/util/cap_alloc	
Capability allocator	1947
l4/re/util/cap_alloc_impl.h	
Capability allocator implementation	2020
l4/re/util/counting_cap_alloc	
Reference-counting capability allocator	2022
l4/re/util/dataspace_svr	??
l4/re/util/debug	??
l4/re/util/env_ns	??
l4/re/util/event	1972
l4/re/util/event_buffer	??
l4/re/util/event_svr	??
l4/re/util/icu_svr	??
l4/re/util/item_alloc	
Item allocator	2025
l4/re/util/kumem_alloc	
Kumem allocator helper	2027
l4/re/util/meta	??
l4/re/util/name_space_svr	??
l4/re/util/object_registry	??
l4/re/util/poll_timeout_kipclock	??
l4/re/util/region_mapping	
Region handling	2029
l4/re/util/region_mapping_svr	??
l4/re/util/region_mapping_svr_2	??
l4/re/util/shared_cap	
Shared_cap / Shared_del_cap	2009
l4/re/util/unique_cap	
Unique_cap / Unique_del_cap	2013
l4/re/util/vcon_svr	??
l4/re/util/video/goos_fb	??
l4/re/util/video/goos_svr	??
l4/re/video/colors	??
l4/re/video/goos	??
l4/re/video/goos-sys.h	
Goos protocol definition	2035
l4/re/video/view	??
l4/shmc/shmc.h	
Shared memory library header file	2036
l4/sigma0/sigma0.h	
Sigma0 interface	2041

l4/sys/___kernel_object_impl.h	
Low-level kernel debugger functions	2044
l4/sys/___kip-32bit.h	??
l4/sys/___kip-64bit.h	??
l4/sys/___ktrace-impl.h	
L4 kernel event tracing	2046
l4/sys/___l4_fpage.h	??
l4/sys/___timeout.h	??
l4/sys/___typeinfo.h	
Type information handling	2048
l4/sys/___vcpu-arm.h	??
l4/sys/___vm-svm.h	??
l4/sys/___vm-vmx.h	??
l4/sys/arm_smccc	??
l4/sys/arm_smccc.h	??
l4/sys/assert.h	
Low-level assert implementation	2251
l4/sys/cache.h	
Cache-consistency functions	2061
l4/sys/capability	
L4::Cap related definitions	2068
l4/sys/compiler.h	
L4 compiler related defines	2072
l4/sys/consts.h	
Common constants	2075
l4/sys/debugger	
The debugger interface specifies common debugging related definitions	2115
l4/sys/debugger.h	
Debugger related definitions	2117
l4/sys/err.h	
Error codes	2124
l4/sys/exception	
Exception C++ interface	2126
l4/sys/factory	
Common factory related definitions	2128
l4/sys/factory.h	
Common factory related definitions	2132
l4/sys/icu	
Interrupt controller	2138
l4/sys/icu.h	
Interrupt controller	2139
l4/sys/iommu	??
l4/sys/ipc.h	
Common IPC interface	2145
l4/sys/ipc_gate	
The C++ IPC gate interface	2154
l4/sys/ipc_gate.h	
The C IPC gate interface	2156
l4/sys/irq	
C++ Irq interface	2159
l4/sys/irq.h	
C Irq interface	1901
l4/sys/kdebug.h	??
l4/sys/kernel_object.h	
Kernel object system calls	2163
l4/sys/kip	2165
l4/sys/kip.h	
Kernel Info Page access functions	2299

l4/sys/kobject	??
l4/sys/ktrace.h	
L4 kernel event tracing	2168
l4/sys/l4int.h	
Fixed sized integer types, generic version	2171
l4/sys/memdesc.h	
Memory description functions	2175
l4/sys/meta	
Meta interface for getting dynamic type information about objects behind capabilities	2179
l4/sys/pager	
Pager and lo_pager C++ interface	2181
l4/sys/platform_control	
Platform control object	2183
l4/sys/platform_control.h	
Platform control object	2185
l4/sys/rcv_endpoint	
The C++ Receive endpoint interface	2189
l4/sys/rcv_endpoint.h	
Receive endpoint C interface	2191
l4/sys/scheduler	
Scheduler object functions	2193
l4/sys/scheduler.h	
Scheduler object functions	2195
l4/sys/semaphore	
Semaphore class definition	2200
l4/sys/semaphore.h	??
l4/sys/smart_capability	
L4::Capability class	2202
l4/sys/task	
Common task related definitions	2206
l4/sys/task.h	
Common task related definitions	2209
l4/sys/thread	
Common thread related definitions	2214
l4/sys/thread.h	
Common thread related definitions	2329
l4/sys/typeinfo_svr	
Type information server template	2219
l4/sys/types.h	
Common L4 ABI Data Types	1819
l4/sys/utcb.h	
UTCB definitions	2221
l4/sys/vcon	
Virtual console interface	2234
l4/sys/vcon.h	
Virtual console interface	2236
l4/sys/vcpu.h	??
l4/sys/vhw.h	
Descriptors for virtual hardware (under UX)	2242
l4/sys/vm	
Virtualization interface	2245
l4/sys/cxx/capability.h	??
l4/sys/cxx/ipc_array	??
l4/sys/cxx/ipc_basics	??
l4/sys/cxx/ipc_client	2083
l4/sys/cxx/ipc_epiface	??
l4/sys/cxx/ipc_iface	
Interface Definition Language	2086

l4/sys/cxx/ipc_legacy	??
l4/sys/cxx/ipc_ret_array	??
l4/sys/cxx/ipc_server	??
l4/sys/cxx/ipc_server_loop	??
l4/sys/cxx/ipc_string	??
l4/sys/cxx/ipc_types	2100
l4/sys/cxx/ipc_varg	??
l4/sys/cxx/smart_capability_1x	2109
l4/sys/cxx/types	2113
l4/util/alloc.h	
Allocator using a bit-array	2246
l4/util/assert.h	
Some useful assert-style macros	2249
l4/util/atomic.h	
Atomic operations header and generic implementations	2254
l4/util/backtrace.h	
Backtrace	2263
l4/util/base64.h	
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01	2265
l4/util/bitops.h	
Bit manipulation functions	2267
l4/util/elf.h	
ELF definition	2272
l4/util/getopt.h	
Getopt	2295
l4/util/keymap.h	
Event to ASCII key mapping	2296
l4/util/kip.h	2297
l4/util/kprintf.h	
Printf using the kernel debugger	2302
l4/util/l4_macros.h	
Some useful generic macros, L4f version	1805
l4/util/list_alloc.h	
Simple list-based allocator	2303
l4/util/llulc.h	??
l4/util/lock.h	
Simple lock implementation	2306
l4/util/mb_info.h	
Multiboot info structure as defined by GRUB	2308
l4/util/parse_cmd.h	
Comfortable command-line parsing	2315
l4/util/rand.h	
Simple Pseudo-Random Number Generator	2316
l4/util/reboot.h	
Machine restarting functions	2318
l4/util/sll.h	
List implementation	2319
l4/util/splitlog2.h	
Split a range in log2 aligned and size-aligned chunks	2323
l4/util/stack.h	
Some helper functions for stack manipulation	2324
l4/util/thread.h	
Low-level Thread Functions	2326
l4/util/util.h	??
l4/vbus/vbus	??
l4/vbus/vbus.h	
Description of the vbus C API	2340
l4/vbus/vbus_generic	??

l4/vbus/vbus_gpio	??
l4/vbus/vbus_gpio-ops.h	??
l4/vbus/vbus_gpio.h	??
l4/vbus/vbus_i2c.h	??
l4/vbus/vbus_inhibitor.h	??
l4/vbus/vbus_interfaces.h	
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs	2343
l4/vbus/vbus_mcspi.h	??
l4/vbus/vbus_pci	??
l4/vbus/vbus_pci-ops.h	??
l4/vbus/vbus_pci.h	??
l4/vbus/vbus_pm-ops.h	??
l4/vbus/vbus_pm.h	??
l4/vbus/vbus_types.h	
This header file contains descriptions of vbus related data types and constants	2345
l4/vbus/vdevice-ops.h	??
l4/vcpu/vcpu	??
l4/vcpu/vcpu.h	??
l4/vcpu/vmx/vmcs.h	??
x86/l4/sys/__kip-arch.h	??
x86/l4/sys/__vcpu-arch.h	??
x86/l4/sys/cache.h	
Cache functions	2067
x86/l4/sys/consts.h	
Common L4 constants, x86 version	2081
x86/l4/sys/ipc-invoke.h	??
x86/l4/sys/ktrace_events.h	??
x86/l4/sys/l4int.h	
Fixed sized integer types, x86 version	2174
x86/l4/sys/linkage.h	
Linkage	1786
x86/l4/sys/segment.h	
Segment handling	1768
x86/l4/sys/syscall-invoke.h	??
x86/l4/sys/utcb.h	
UTCB definitions for X86	2231
x86/l4/sys/vm.h	??
x86/l4/util/apic.h	
APIC for X86	1720
x86/l4/util/bitops_arch.h	
X86 bit manipulation functions	1795
x86/l4/util/cpu.h	
CPU related functions	1802
x86/l4/util/idt.h	
IDT related functions	1728
x86/l4/util/irq.h	
Some PIC and hardware interrupt related functions	1899
x86/l4/util/l4_macros.h	
Main function	1806
x86/l4/util/mbi_argv.h	
Command line handling	1809
x86/l4/util/perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1735
x86/l4/util/port_io.h	
X86 port I/O	1775
x86/l4/util/rdtsc.h	
Time stamp counter related functions	1746

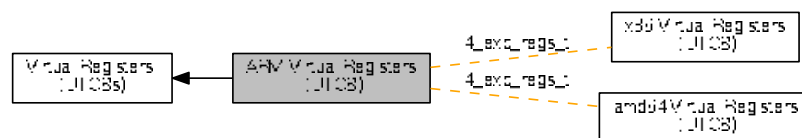
x86/l4/util/ spin.h	
Spinning for x86	1752
x86/l4/util/ stack_impl.h	
Stack utilities for x86	1813
x86/l4/util/ util.h	
Utilities for x86	1756
x86/l4f/l4/sys/ ipc-l42-gcc3-nopic.h	??
x86/l4f/l4/sys/ ipc.h	
L4 IPC System Calls, x86	2153
x86/l4f/l4/sys/ segment.h	
L4f specific segment manipulation	1766
x86/l4f/l4/util/ port_io.h	
Port I/O functions	1772
x86/l4f/l4/util/ setjmp.h	
Inter-thread setjmp/longjmp	1782

Chapter 12

Module Documentation

12.1 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Typedefs

- typedef struct [l4_exc_regs_t](#) [l4_exc_regs_t](#)
UTCB structure for exceptions.

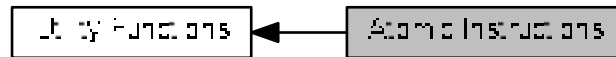
Enumerations

- enum [L4_utcb_consts_arm](#)
UTCB constants for ARM.

12.1.1 Detailed Description

12.2 Atomic Instructions

Collaboration diagram for Atomic Instructions:



Files

- file [atomic.h](#)
atomic operations header and generic implementations

Functions

- int [l4util_cmpxchg64](#) (volatile [l4_uint64_t](#) *dest, [l4_uint64_t](#) cmp_val, [l4_uint64_t](#) new_val)
Atomic compare and exchange (64 bit version)
- int [l4util_cmpxchg32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) cmp_val, [l4_uint32_t](#) new_val)
Atomic compare and exchange (32 bit version)
- int [l4util_cmpxchg16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) cmp_val, [l4_uint16_t](#) new_val)
Atomic compare and exchange (16 bit version)
- int [l4util_cmpxchg8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) cmp_val, [l4_uint8_t](#) new_val)
Atomic compare and exchange (8 bit version)
- int [l4util_cmpxchg](#) (volatile [l4_umword_t](#) *dest, [l4_umword_t](#) cmp_val, [l4_umword_t](#) new_val)
Atomic compare and exchange (machine wide fields)
- [l4_uint32_t](#) [l4util_xchg32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
Atomic exchange (32 bit version)
- [l4_uint16_t](#) [l4util_xchg16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
Atomic exchange (16 bit version)
- [l4_uint8_t](#) [l4util_xchg8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
Atomic exchange (8 bit version)
- [l4_umword_t](#) [l4util_xchg](#) (volatile [l4_umword_t](#) *dest, [l4_umword_t](#) val)
Atomic exchange (machine wide fields)
- void [l4util_atomic_add](#) (volatile long *dest, long val)
Atomic add.
- void [l4util_atomic_inc](#) (volatile long *dest)
Atomic increment.

Atomic add/sub/and/or (8,16,32 bit version) without result

- void `l4util_add8` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- void `l4util_add16` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- void `l4util_add32` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)
- void `l4util_sub8` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- void `l4util_sub16` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- void `l4util_sub32` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)
- void `l4util_and8` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- void `l4util_and16` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- void `l4util_and32` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)
- void `l4util_or8` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- void `l4util_or16` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- void `l4util_or32` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)

Atomic add/sub/and/or operations (8,16,32 bit) with result

- `l4_uint8_t l4util_add8_res` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_add16_res` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_add32_res` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_sub8_res` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_sub16_res` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_sub32_res` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_and8_res` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_and16_res` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_and32_res` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_or8_res` (volatile `l4_uint8_t` *dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_or16_res` (volatile `l4_uint16_t` *dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_or32_res` (volatile `l4_uint32_t` *dest, `l4_uint32_t` val)

Atomic inc/dec (8,16,32 bit) without result

- void `l4util_inc8` (volatile `l4_uint8_t` *dest)
- void `l4util_inc16` (volatile `l4_uint16_t` *dest)
- void `l4util_inc32` (volatile `l4_uint32_t` *dest)
- void `l4util_dec8` (volatile `l4_uint8_t` *dest)
- void `l4util_dec16` (volatile `l4_uint16_t` *dest)
- void `l4util_dec32` (volatile `l4_uint32_t` *dest)

Atomic inc/dec (8,16,32 bit) with result

- `l4_uint8_t l4util_inc8_res` (volatile `l4_uint8_t` *dest)
- `l4_uint16_t l4util_inc16_res` (volatile `l4_uint16_t` *dest)
- `l4_uint32_t l4util_inc32_res` (volatile `l4_uint32_t` *dest)
- `l4_uint8_t l4util_dec8_res` (volatile `l4_uint8_t` *dest)
- `l4_uint16_t l4util_dec16_res` (volatile `l4_uint16_t` *dest)
- `l4_uint32_t l4util_dec32_res` (volatile `l4_uint32_t` *dest)

12.2.1 Detailed Description

12.2.2 Function Documentation

12.2.2.1 l4util_add8()

```
void l4util_add8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val ) [inline]
```

Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 424 of file [atomic.h](#).

12.2.2.2 l4util_add8_res()

```
l4_uint8_t l4util_add8_res (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val ) [inline]
```

Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Returns

res

Definition at line 476 of file [atomic.h](#).

12.2.2.3 l4util_atomic_add()

```
void l4util_atomic_add (  
    volatile long * dest,  
    long val ) [inline]
```

Atomic add.

Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add

Definition at line 436 of file [atomic.h](#).

12.2.2.4 l4util_atomic_inc()

```
void l4util_atomic_inc (
    volatile long * dest ) [inline]
```

Atomic increment.

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 379 of file [atomic.h](#).

12.2.2.5 l4util_cmpxchg()

```
int l4util_cmpxchg (
    volatile l4_umword_t * dest,
    l4_umword_t cmp_val,
    l4_umword_t new_val ) [inline]
```

Atomic compare and exchange (machine wide fields)

Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

Returns

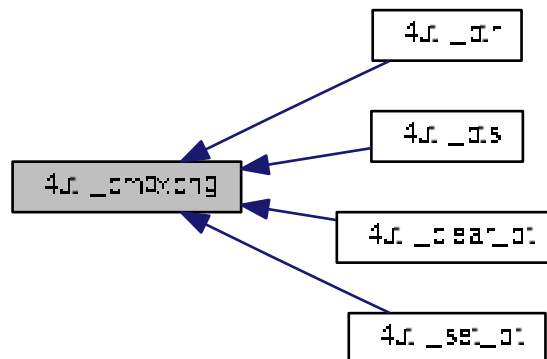
0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 335 of file [atomic.h](#).

Referenced by [l4util_btr\(\)](#), [l4util_bts\(\)](#), [l4util_clear_bit\(\)](#), and [l4util_set_bit\(\)](#).

Here is the caller graph for this function:



12.2.2.6 l4util_cmpxchg16()

```

int l4util_cmpxchg16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t cmp_val,
    l4_uint16_t new_val ) [inline]
  
```

Atomic compare and exchange (16 bit version)

Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 319 of file [atomic.h](#).

12.2.2.7 l4util_cmpxchg32()

```
int l4util_cmpxchg32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t cmp_val,
    l4_uint32_t new_val ) [inline]
```

Atomic compare and exchange (32 bit version)

Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 311 of file [atomic.h](#).

12.2.2.8 l4util_cmpxchg64()

```
int l4util_cmpxchg64 (
    volatile l4_uint64_t * dest,
    l4_uint64_t cmp_val,
    l4_uint64_t new_val ) [inline]
```

Atomic compare and exchange (64 bit version)

Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

Returns

0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 303 of file [atomic.h](#).

12.2.2.9 l4util_cmpxchg8()

```
int l4util_cmpxchg8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t cmp_val,
    l4_uint8_t new_val ) [inline]
```

Atomic compare and exchange (8 bit version)

Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 327 of file [atomic.h](#).

12.2.2.10 l4util_inc8()

```
void l4util_inc8 (
    volatile l4_uint8_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 367 of file [atomic.h](#).

12.2.2.11 l4util_inc8_res()

```
l4_uint8_t l4util_inc8_res (
    volatile l4_uint8_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Returns

res

Definition at line 396 of file [atomic.h](#).

12.2.2.12 l4util_xchg()

```
l4_umword_t l4util_xchg (
    volatile l4_umword_t * dest,
    l4_umword_t val ) [inline]
```

Atomic exchange (machine wide fields)

Parameters

<i>dest</i>	destination operand
<i>val</i>	new value for dest

Returns

old value at destination

Definition at line 361 of file [atomic.h](#).

12.2.2.13 l4util_xchg16()

```
l4_uint16_t l4util_xchg16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t val ) [inline]
```

Atomic exchange (16 bit version)

Parameters

<i>dest</i>	destination operand
<i>val</i>	new value for dest

Returns

old value at destination

Definition at line 349 of file [atomic.h](#).

12.2.2.14 l4util_xchg32()

```
l4_uint32_t l4util_xchg32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t val ) [inline]
```

Atomic exchange (32 bit version)

Parameters

<i>dest</i>	destination operand
<i>val</i>	new value for dest

Returns

old value at destination

Definition at line 343 of file [atomic.h](#).

12.2.2.15 l4util_xchg8()

```
l4_uint8_t l4util_xchg8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val ) [inline]
```

Atomic exchange (8 bit version)

Parameters

<i>dest</i>	destination operand
<i>val</i>	new value for dest

Returns

old value at destination

Definition at line 355 of file [atomic.h](#).

12.3 Auxiliary data

Collaboration diagram for Auxiliary data:



Data Structures

- struct [l4re_aux_t](#)
Auxiliary descriptor.

Typedefs

- typedef struct [l4re_aux_t](#) [l4re_aux_t](#)
Auxiliary descriptor.

Enumerations

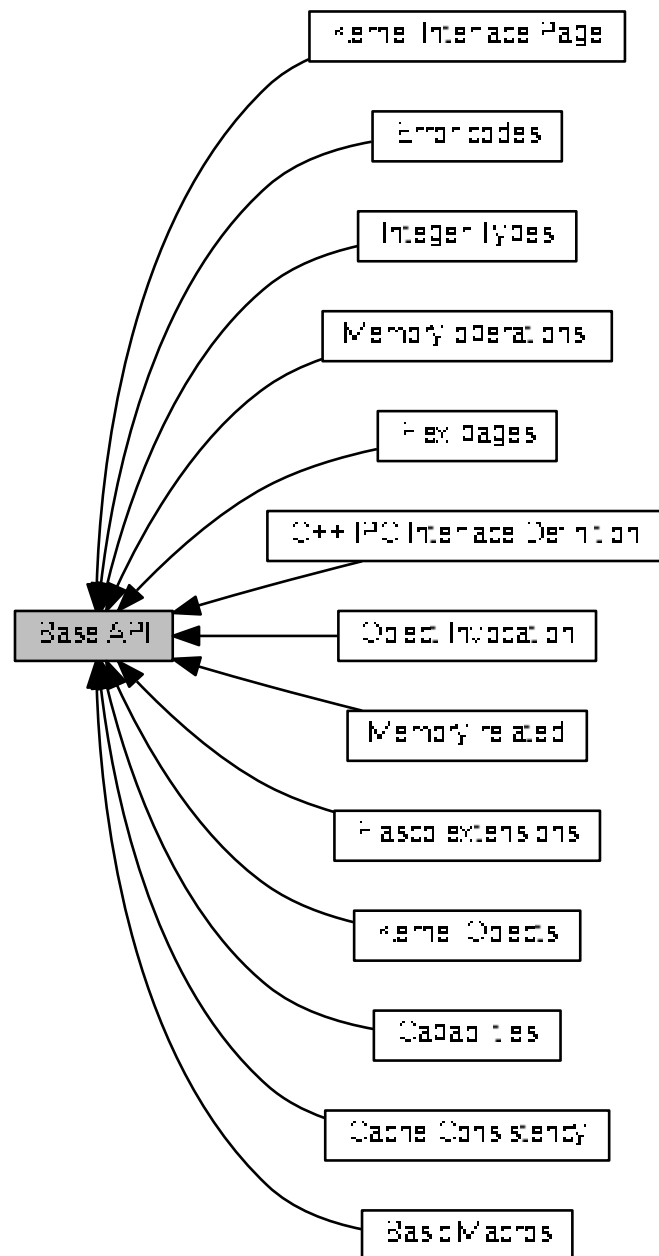
- enum [l4re_aux_ldr_flags_t](#)
Flags for program loading.

12.3.1 Detailed Description

12.4 Base API

Interfaces for all kinds of base functionality.

Collaboration diagram for Base API:



Modules

- [Basic Macros](#)

- [L4 standard macros for header files, function definitions, and public APIs etc.](#)
- [C++ IPC Interface Definition.](#)
APIs for defining IPC interfaces using C++ as language.
- [Cache Consistency](#)
Various functions for cache consistency.
- [Capabilities](#)
C interface for capabilities.
- [Error codes](#)
Common error codes.
- [Fiasco extensions](#)
Kernel debugger extensions of the Fiasco L4 implementation.
- [Flex pages](#)
Flex-page related API.
- [Integer Types](#)
- [Kernel Interface Page](#)
Kernel Interface Page.
- [Kernel Objects](#)
API of kernel objects.
- [Memory operations.](#)
Operations for memory access.
- [Memory related](#)
Memory related constants, data types and functions.
- [Object Invocation](#)
API for L4 object invocation.

Files

- file [cache.h](#)
Cache-consistency functions.
- file [compiler.h](#)
L4 compiler related defines.
- file [consts.h](#)
Common constants.
- file [debugger.h](#)
Debugger related definitions.
- file [factory.h](#)
Common factory related definitions.
- file [icu](#)
Interrupt controller.
- file [icu.h](#)
Interrupt controller.
- file [ipc.h](#)
Common IPC interface.
- file [irq.h](#)
C Irq interface.
- file [kip](#)
- file [kip.h](#)
Kernel Info Page access functions.
- file [memdesc.h](#)
Memory description functions.

- file [types.h](#)
Common L4 ABI Data Types.
- file [vhw.h](#)
Descriptors for virtual hardware (under UX).
- file [consts.h](#)
Common L4 constants, arm version.
- file [consts.h](#)
Common L4 constants, amd64 version.
- file [ipc.h](#)
L4 IPC System Calls, x86.
- file [consts.h](#)
Common L4 constants, x86 version.

12.4.1 Detailed Description

Interfaces for all kinds of base functionality.

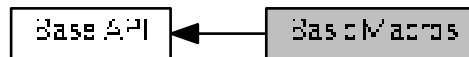
Some notes on Inter Process Communication (IPC)

IPC in L4 is always synchronous and unbuffered: a message is transferred from the sender to the recipient if and only if the recipient has invoked a corresponding IPC operation. The sender blocks until this happens or a timeout specified by the sender elapsed without the destination becoming ready to receive.

12.5 Basic Macros

[L4](#) standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:



Macros

- `#define L4_ALWAYS_INLINE`
L4 Inline function attribute.
- `#define __END_DECLS`
End section with C types and functions.
- `#define EXTERN_C_BEGIN`
Start section with C types and functions.
- `#define EXTERN_C_END`
End section with C types and functions.
- `#define EXTERN_C`
Mark C types and functions.
- `#define L4_NOTHROW`
Mark a function declaration and definition as never throwing an exception.
- `#define L4_HIDDEN`
Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.
- `#define L4_NORETURN`
Noreturn function attribute.
- `#define L4_NOINSTRUMENT`
No instrumentation function attribute.
- `#define L4_LIKELY(x)`
Expression is likely to execute.
- `#define L4_UNLIKELY(x)`
Expression is unlikely to execute.
- `#define L4_STICKY(x)`
Mark symbol sticky (even not there)
- `#define L4_DEPRECATED(s)`
Mark symbol deprecated.
- `#define L4_stringify_helper(x)`
stringify helper.
- `#define L4_stringify(x)`
stringify.
- `#define L4_CV`
Define calling convention.
- `#define L4_CV`
Define calling convention.
- `#define L4_CV __attribute__((regparm(0)))`
Define calling convention.

Functions

- void [l4_barrier](#) (void)
Memory barrier.
- void [l4_mb](#) (void)
Memory barrier.
- void [l4_wmb](#) (void)
Write memory barrier.

12.5.1 Detailed Description

[L4](#) standard macros for header files, function definitions, and public APIs etc.

Include File

```
#include <l4/sys/compiler.h>
```

12.5.2 Macro Definition Documentation

12.5.2.1 L4_ALWAYS_INLINE

```
#define L4_ALWAYS_INLINE
```

[L4](#) Inline function attribute.

Always inline a function

Definition at line [67](#) of file [compiler.h](#).

12.5.2.2 L4_HIDDEN

```
#define L4_HIDDEN
```

Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.

This attribute is intended for functions, data, and data types that shall never be visible outside of a library. In particular, for shared libraries this may result in much faster code within the library and short linking times.

```
class L4\_HIDDEN My_class  
{  
    ...  
};  
  
int L4\_HIDDEN function(void);  
  
int L4\_HIDDEN global_data; // global data is not recommended
```

Definition at line [212](#) of file [compiler.h](#).

12.5.2.3 L4_NOTHROW

```
#define L4_NOTHROW
```

Mark a function declaration and definition as never throwing an exception.

(Also for C code).

This macro shall be used to mark C and C++ functions that never throw any exception. Note that also C functions may throw exceptions according to the compilers ABI and shall be marked with L4_NOTHROW if they never do. In C++ this is equivalent to `throw()`.

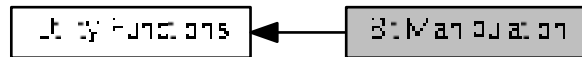
```
int foo() L4_NOTHROW;  
...  
int foo() L4_NOTHROW  
{  
    ...  
    return result;  
}
```

Definition at line 185 of file [compiler.h](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), [l4_round_page\(\)](#), [l4_sleep_forever\(\)](#), [l4_trunc_page\(\)](#), [l4_trunc_size\(\)](#), [l4_vm_vmx_field_len\(\)](#), [l4vcpu_irq_disable_save\(\)](#), and [l4virtio_get_feature\(\)](#).

12.6 Bit Manipulation

Collaboration diagram for Bit Manipulation:



Files

- file [bitops.h](#)
bit manipulation functions

Functions

- void [l4util_set_bit](#) (int b, volatile [l4_umword_t](#) *dest)
Set bit in memory.
- void [l4util_clear_bit](#) (int b, volatile [l4_umword_t](#) *dest)
Clear bit in memory.
- void [l4util_complement_bit](#) (int b, volatile [l4_umword_t](#) *dest)
Complement bit in memory.
- int [l4util_test_bit](#) (int b, const volatile [l4_umword_t](#) *dest)
Test bit (return value of bit)
- int [l4util_bts](#) (int b, volatile [l4_umword_t](#) *dest)
Bit test and set.
- int [l4util_btr](#) (int b, volatile [l4_umword_t](#) *dest)
Bit test and reset.
- int [l4util_btc](#) (int b, volatile [l4_umword_t](#) *dest)
Bit test and complement.
- int [l4util_bsr](#) ([l4_umword_t](#) word)
Bit scan reverse.
- int [l4util_bsf](#) ([l4_umword_t](#) word)
Bit scan forward.
- int [l4util_find_first_set_bit](#) (const void *dest, [l4_size_t](#) size)
Find the first set bit in a memory region.
- int [l4util_find_first_zero_bit](#) (const void *dest, [l4_size_t](#) size)
Find the first zero bit in a memory region.
- int [l4util_next_power2](#) (const unsigned long val)
Find the next power of 2 for a given number.

12.6.1 Detailed Description

12.6.2 Function Documentation

12.6.2.1 l4util_bsf()

```
int l4util_bsf (
    l4_umword_t word ) [inline]
```

Bit scan forward.

Parameters

<i>word</i>	value (machine size)
-------------	----------------------

Returns

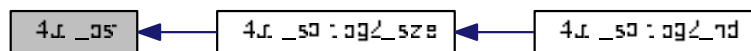
index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 316 of file [bitops.h](#).

Referenced by [l4util_splitlog2_size\(\)](#).

Here is the caller graph for this function:



12.6.2.2 l4util_bsr()

```
int l4util_bsr (
    l4_umword_t word ) [inline]
```

Bit scan reverse.

Parameters

<i>word</i>	value (machine size)
-------------	----------------------

Returns

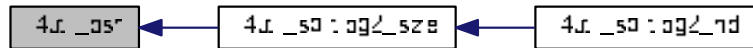
index of most significant set bit in word, -1 if no bit is set (word == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(word))

Definition at line 299 of file [bitops.h](#).

Referenced by [l4util_splitlog2_size\(\)](#).

Here is the caller graph for this function:



12.6.2.3 l4util_btc()

```
int l4util_btc (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Bit test and complement.

Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Returns

Old value of bit *b*.

Complement bit *b* and return old value.

Definition at line 394 of file [bitops.h](#).

12.6.2.4 l4util_btr()

```
int l4util_btr (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Bit test and reset.

Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Returns

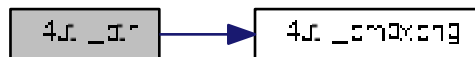
Old value of bit *b*.

Reset bit *b* and return old value.

Definition at line 278 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:

**12.6.2.5 l4util_bts()**

```
int l4util_bts (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Bit test and set.

Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Returns

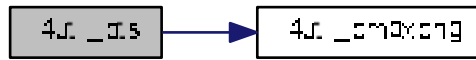
Old value of bit *b*.

Set the *b* bit of *dest* to 1 and return the old value.

Definition at line 256 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



12.6.2.6 l4util_clear_bit()

```
void l4util_clear_bit (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Clear bit in memory.

Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 226 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



12.6.2.7 l4util_complement_bit()

```
void l4util_complement_bit (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Complement bit in memory.

Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 359 of file [bitops.h](#).

12.6.2.8 l4util_find_first_set_bit()

```
int l4util_find_first_set_bit (
    const void * dest,
    l4_size_t size ) [inline]
```

Find the first set bit in a memory region.

Parameters

<i>dest</i>	bit string
<i>size</i>	size of string in bits (must be a multiple of 32!)

Returns

number of the first set bit, >= size if no bit is set

Definition at line 400 of file [bitops.h](#).

12.6.2.9 l4util_find_first_zero_bit()

```
int l4util_find_first_zero_bit (
    const void * dest,
    l4_size_t size ) [inline]
```

Find the first zero bit in a memory region.

Parameters

<i>dest</i>	bit string
<i>size</i>	size of string in bits (must be a multiple of 32!)

Returns

number of the first zero bit, >= size if no bit is set

Definition at line 333 of file [bitops.h](#).

12.6.2.10 l4util_next_power2()

```
int l4util_next_power2 (
    const unsigned long val ) [inline]
```

Find the next power of 2 for a given number.

Parameters

<i>val</i>	initial value
------------	---------------

Returns

next-highest power of 2

Definition at line 373 of file [bitops.h](#).

12.6.2.11 l4util_set_bit()

```
void l4util_set_bit (
    int b,
    volatile l4_umword_t * dest ) [inline]
```

Set bit in memory.

Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 207 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



12.6.2.12 l4util_test_bit()

```
int l4util_test_bit (  
    int b,  
    const volatile l4_umword_t * dest )    [inline]
```

Test bit (return value of bit)

Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

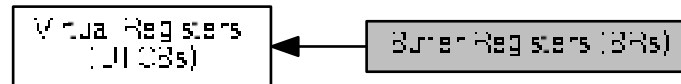
Returns

Value of bit *b*.

Definition at line 244 of file [bitops.h](#).

12.7 Buffer Registers (BRs)

Collaboration diagram for Buffer Registers (BRs):



Data Structures

- struct [l4_buf_regs_t](#)
Encapsulation of the buffer-registers block in the UTCB.

Typedefs

- typedef struct [l4_buf_regs_t](#) [l4_buf_regs_t](#)
Encapsulation of the buffer-registers block in the UTCB.

Enumerations

- enum [l4_buffer_desc_consts_t](#) { [L4_BDR_MEM_SHIFT](#) = 0, [L4_BDR_IO_SHIFT](#) = 5, [L4_BDR_OBJ_SHIFT](#) = 10 }
Constants for buffer descriptors.

Functions

- void [l4_utcb_inherit_fpu](#) (int switch_on) [L4_NOTHROW](#)
Enable or disable inheritance of FPU state to receiver.

12.7.1 Detailed Description

12.7.2 Enumeration Type Documentation

12.7.2.1 [l4_buffer_desc_consts_t](#)

```
enum l4\_buffer\_desc\_consts\_t
```

Constants for buffer descriptors.

Enumerator

L4_BDR_MEM_SHIFT	Bit offset for the memory-buffer index.
L4_BDR_IO_SHIFT	Bit offset for the IO-buffer index.
L4_BDR_OBJ_SHIFT	Bit offset for the capability-buffer index.

Definition at line 219 of file [consts.h](#).

12.8 C++ Exceptions

Collaboration diagram for C++ Exceptions:



Files

- file [exceptions](#)
Base exceptions.
- file [std_exc_io](#)
Base exceptions std stream operator.

Data Structures

- class [L4::Exception_tracer](#)
Back-trace support for exceptions.
- class [L4::Base_exception](#)
Base class for all exceptions, thrown by the [L4Re](#) framework.
- class [L4::Runtime_error](#)
Exception for an abstract runtime error.
- class [L4::Out_of_memory](#)
Exception signalling insufficient memory.
- class [L4::Element_already_exists](#)
Exception for duplicate element insertions.
- class [L4::Unknown_error](#)
Exception for an unknown condition.
- class [L4::Element_not_found](#)
Exception for a failed lookup (element not found).
- class [L4::Invalid_capability](#)
Indicates that an invalid object was invoked.
- class [L4::Com_error](#)
Error conditions during IPC.
- class [L4::Bounds_error](#)
Access out of bounds.

Macros

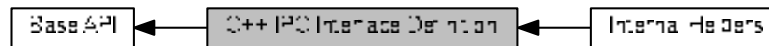
- `#define L4_CXX_EXCEPTION_BACKTRACE 20`
Number of instruction pointers in backtrace.

12.8.1 Detailed Description

12.9 C++ IPC Interface Definition.

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition.:



Modules

- [Internal Helpers](#)

Namespaces

- [L4::Typeid](#)
Definition of interface data-type helpers.

12.9.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

12.10 CPU related functions

Collaboration diagram for CPU related functions:



Functions

- int [l4util_cpu_has_cpuid](#) (void)
Check whether the CPU supports the "cpuid" instruction.
- unsigned int [l4util_cpu_capabilities](#) (void)
Returns the CPU capabilities if the "cpuid" instruction is available.
- unsigned int [l4util_cpu_capabilities_noccheck](#) (void)
Returns the CPU capabilities.
- void [l4util_cpu_cpuid](#) (unsigned long mode, unsigned long *eax, unsigned long *ebx, unsigned long *ecx, unsigned long *edx)
Generic CPUID access function.

12.10.1 Detailed Description

12.10.2 Function Documentation

12.10.2.1 [l4util_cpu_capabilities\(\)](#)

```
unsigned int l4util_cpu_capabilities (
    void ) [inline]
```

Returns the CPU capabilities if the "cpuid" instruction is available.

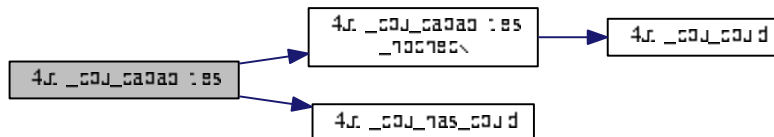
Returns

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 97 of file [cpu.h](#).

References [EXTERN_C_END](#), [l4util_cpu_capabilities_nocheck\(\)](#), and [l4util_cpu_has_cpuid\(\)](#).

Here is the call graph for this function:

**12.10.2.2 l4util_cpu_capabilities_nocheck()**

```

unsigned int l4util_cpu_capabilities_nocheck (
    void ) [inline]
  
```

Returns the CPU capabilities.

Returns

CPU capabilities.

Definition at line 86 of file [cpu.h](#).

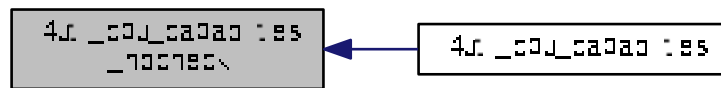
References [l4util_cpu_cpuid\(\)](#).

Referenced by [l4util_cpu_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.10.2.3 l4util_cpu_has_cpuid()

```
int l4util_cpu_has_cpuid (
    void ) [inline]
```

Check whether the CPU supports the "cpuid" instruction.

Returns

1 if it has, 0 if it has not

Definition at line 66 of file [cpu.h](#).

Referenced by [l4util_cpu_capabilities\(\)](#).

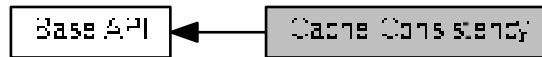
Here is the caller graph for this function:



12.11 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:



Functions

- int [l4_cache_clean_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache clean a range in D-cache.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory.

12.11.1 Detailed Description

Various functions for cache consistency.

Include File

```
#include <l4/sys/cache.h>
```

12.11.2 Function Documentation

12.11.2.1 l4_cache_clean_data()

```
int l4_cache_clean_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache clean a range in D-cache.

Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines in the range but leaves them in the cache.

Examples:

[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line [84](#) of file [cache.h](#).

12.11.2.2 `l4_cache_coherent()`

```
int l4_cache_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent between I-cache and D-cache.

Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Definition at line [108](#) of file [cache.h](#).

12.11.2.3 `l4_cache_dma_coherent()`

```
int l4_cache_dma_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent for use with external memory.

Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Definition at line 116 of file [cache.h](#).

12.11.2.4 l4_cache_flush_data()

```
int l4_cache_flush_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache flush a range.

Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines and invalidates all cache entries in the range.

Definition at line 92 of file [cache.h](#).

12.11.2.5 l4_cache_inv_data()

```
int l4_cache_inv_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache invalidate a range.

Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Invalidates all cache entries in the range but does not necessarily write back dirty cache lines.

Note

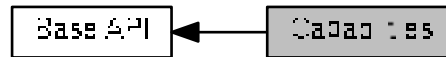
Implementations may choose to write back dirty lines nonetheless if this is more efficient.

Definition at line 100 of file [cache.h](#).

12.12 Capabilities

C interface for capabilities.

Collaboration diagram for Capabilities:



Typedefs

- typedef unsigned long [l4_cap_idx_t](#)
L4 Capability selector Type.

Enumerations

- enum [l4_cap_consts_t](#) { [L4_CAP_SHIFT](#), [L4_CAP_SIZE](#), [L4_CAP_MASK](#), [L4_INVALID_CAP](#) }
Constants related to capability selectors.
- enum [l4_default_caps_t](#) {
[L4_BASE_TASK_CAP](#), [L4_BASE_FACTORY_CAP](#), [L4_BASE_THREAD_CAP](#), [L4_BASE_PAGER_CAP](#),
[L4_BASE_LOG_CAP](#), [L4_BASE_ICU_CAP](#), [L4_BASE_SCHEDULER_CAP](#), [L4_BASE_IOMMU_CAP](#),
[L4_BASE_DEBUGGER_CAP](#), [L4_BASE_ARM_SMCCC_CAP](#), [L4_BASE_CAPS_LAST](#) = [L4_BASE_CAPS](#)
[PS_LAST_P1](#) - 1 }
Default capabilities setup for the initial tasks.

Functions

- unsigned [l4_is_invalid_cap](#) ([l4_cap_idx_t](#) c) [L4_NOTHROW](#)
Test if a capability selector is the invalid capability.
- unsigned [l4_is_valid_cap](#) ([l4_cap_idx_t](#) c) [L4_NOTHROW](#)
Test if a capability selector is a valid selector.
- unsigned [l4_capability_equal](#) ([l4_cap_idx_t](#) c1, [l4_cap_idx_t](#) c2) [L4_NOTHROW](#)
Test if two capability selectors are equal.

12.12.1 Detailed Description

C interface for capabilities.

Add

```
#include <l4/sys/types.h>
#include <l4/sys/consts.h>
```

to your code to use the functions and definitions explained here.

12.12.2 Enumeration Type Documentation

12.12.2.1 l4_cap_consts_t

```
enum l4_cap_consts_t
```

Constants related to capability selectors.

Enumerator

L4_CAP_SHIFT	Capability index shift.
L4_CAP_SIZE	Offset of two consecutive capability selectors.
L4_CAP_MASK	Mask to get only the relevant bits of an l4_cap_idx_t.
L4_INVALID_CAP	Invalid capability selector.

Definition at line 128 of file [consts.h](#).

12.12.2.2 l4_default_caps_t

```
enum l4_default_caps_t
```

Default capabilities setup for the initial tasks.

These capability selectors are setup per default by the micro kernel for the two initial tasks, the Root-Pager (Sigma0) and the Root-Task (Moe).

Attention

This constants do not have any particular meaning for applications started by Moe, see [Initial Environment](#) for this kind of information.

See also

[Initial Environment](#) for information useful for normal user applications.

Enumerator

L4_BASE_TASK_CAP	Capability selector for the current task.
L4_BASE_FACTORY_CAP	Capability selector for the factory.
L4_BASE_THREAD_CAP	Capability selector for the first thread.
L4_BASE_PAGER_CAP	Capability selector for the pager gate. For Sigma0, the pager is not present since it never raises page faults. For Moe, the pager is set to Sigma0.
L4_BASE_LOG_CAP	Capability selector for the log object. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ICU_CAP	Capability selector for the base icu object.

Enumerator

L4_BASE_SCHEDULER_CAP	Capability selector for the scheduler cap.
L4_BASE_IOMMU_CAP	Capability selector for the IO-MMU cap. Present if the microkernel detected an IO-MMU.
L4_BASE_DEBUGGER_CAP	Capability selector for the debugger cap. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ARM_SMCCC_CAP	Capability selector for the ARM SMCCC cap. Present if the microkernel detected an ARM SMC capable trusted execution environment.
L4_BASE_CAPS_LAST	Last capability index used for base capabilities.

Definition at line 240 of file [consts.h](#).

12.12.3 Function Documentation

12.12.3.1 l4_capability_equal()

```
unsigned l4_capability_equal (
    l4_cap_idx_t c1,
    l4_cap_idx_t c2 ) [inline]
```

Test if two capability selectors are equal.

Parameters

<i>c1</i>	Capability
<i>c2</i>	Capability

Return values

0	The given capabilities are not in the same slot.
1	The given capabilities are in the same slot.

Definition at line 400 of file [types.h](#).

References [L4_CAP_SHIFT](#).

12.12.3.2 l4_is_invalid_cap()

```
unsigned l4_is_invalid_cap (
    l4_cap_idx_t c ) [inline]
```

Test if a capability selector is the invalid capability.

Parameters

<code>c</code>	Capability selector
----------------	---------------------

Return values

<code>0</code>	The capability selector is not the invalid capability.
<code>>0</code>	The capability selector is the invalid capability.

Examples:

[examples/libs/l4re/c/main.c](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 392 of file [types.h](#).

12.12.3.3 `l4_is_valid_cap()`

```
unsigned l4_is_valid_cap (
    l4_cap_idx_t c ) [inline]
```

Test if a capability selector is a valid selector.

Parameters

<code>c</code>	Capability selector
----------------	---------------------

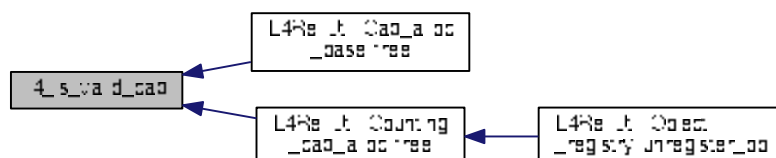
Return values

<code>0</code>	The capability selector is not valid.
<code>>0</code>	The capability selector is valid.

Definition at line 396 of file [types.h](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), and [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free\(\)](#).

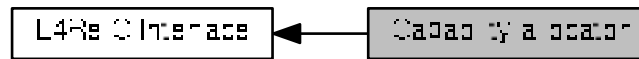
Here is the caller graph for this function:



12.13 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:



Functions

- [l4re_cap_idx_t l4re_util_cap_alloc \(void\) L4_NOTHROW](#)
Get free capability index at capability allocator.
- [void l4re_util_cap_free \(l4re_cap_idx_t cap\) L4_NOTHROW](#)
Return capability index to capability allocator.
- [void l4re_util_cap_free_um \(l4re_cap_idx_t cap\) L4_NOTHROW](#)
Return capability index to capability allocator, and unmaps the object.
- [long l4re_util_cap_last \(void\) L4_NOTHROW](#)
Return last capability index the allocator can return.

12.13.1 Detailed Description

Capability allocator C interface.

12.13.2 Function Documentation

12.13.2.1 l4re_util_cap_last()

```
long l4re_util_cap_last (
    void )
```

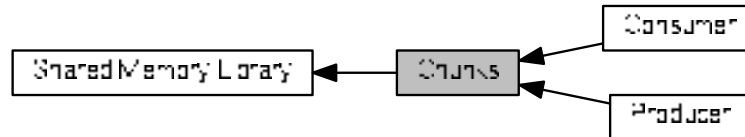
Return last capability index the allocator can return.

Returns

last/biggest capability index the allocator can return

12.14 Chunks

Collaboration diagram for Chunks:



Modules

- [Consumer](#)
- [Producer](#)

Functions

- long [l4shmc_add_chunk](#) (l4shmc_area_t *shmarea, const char *chunk_name, [l4_umword_t](#) chunk_capacity, l4shmc_chunk_t *chunk)
Add a chunk in the shared memory area.
- long [l4shmc_get_chunk](#) (l4shmc_area_t *shmarea, const char *chunk_name, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area.
- long [l4shmc_get_chunk_to](#) (l4shmc_area_t *shmarea, const char *chunk_name, [l4_umword_t](#) timeout_ms, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area, with timeout.
- long [l4shmc_iterate_chunk](#) (l4shmc_area_t *shmarea, const char **chunk_name, long offs)
Iterate over names of all existing chunks.
- void * [l4shmc_chunk_ptr](#) (l4shmc_chunk_t *chunk)
Get data pointer to chunk.
- long [l4shmc_chunk_capacity](#) (l4shmc_chunk_t *chunk)
Get capacity of a chunk.
- l4shmc_signal_t * [l4shmc_chunk_signal](#) (l4shmc_chunk_t *chunk)
Get the signal of a chunk.

12.14.1 Detailed Description

12.14.2 Function Documentation

12.14.2.1 l4shmc_add_chunk()

```

long l4shmc_add_chunk (
    l4shmc_area_t * shmarea,
    const char * chunk_name,
    l4\_umword\_t chunk_capacity,
    l4shmc_chunk_t * chunk )
  
```

Add a chunk in the shared memory area.

Parameters

	<i>shmarea</i>	The shared memory area to put the chunk in.
	<i>chunk_name</i>	Name of the chunk.
	<i>chunk_capacity</i>	Capacity for payload of the chunk in bytes.
out	<i>chunk</i>	Chunk structure to fill in.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.14.2.2 l4shmc_chunk_capacity()

```
long l4shmc_chunk_capacity (
    l4shmc_chunk_t * chunk ) [inline]
```

Get capacity of a chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	Success.
<0	Error.

12.14.2.3 l4shmc_chunk_ptr()

```
void* l4shmc_chunk_ptr (
    l4shmc_chunk_t * chunk ) [inline]
```

Get data pointer to chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.14.2.4 l4shmc_chunk_signal()

```
l4shmc_signal_t* l4shmc_chunk_signal (
    l4shmc_chunk_t * chunk ) [inline]
```

Get the signal of a chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	No signal has been registered with this chunk.
>0	Pointer to signal otherwise.

12.14.2.5 l4shmc_get_chunk()

```
long l4shmc_get_chunk (
    l4shmc_area_t * shmarea,
    const char * chunk_name,
    l4shmc_chunk_t * chunk ) [inline]
```

Get chunk out of shared memory area.

Parameters

	<i>shmarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
out	<i>chunk</i>	Chunk data structure to fill.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.14.2.6 l4shmc_get_chunk_to()

```
long l4shmc_get_chunk_to (
    l4shmc_area_t * shmarea,
    const char * chunk_name,
    l4_umword_t timeout_ms,
    l4shmc_chunk_t * chunk )
```

Get chunk out of shared memory area, with timeout.

Parameters

	<i>shmarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	<i>chunk</i>	Chunk data structure to fill.

Return values

0	Success.
<0	Error.

12.14.2.7 l4shmc_iterate_chunk()

```
long l4shmc_iterate_chunk (
    l4shmc_area_t * shmarea,
    const char ** chunk_name,
    long offs )
```

Iterate over names of all existing chunks.

Parameters

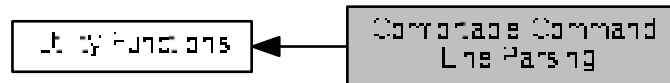
<i>shmarea</i>	Shared memory area.
<i>chunk_name</i>	Where the name of the current chunk will be stored
<i>offs</i>	0 to start iteration, return value of previous call to l4shmc_iterate_chunk() to get next chunk

Return values

0	No more chunks available.
<0	Error.
>0	Iterator value for the next call.

12.15 Comfortable Command Line Parsing

Collaboration diagram for Comfortable Command Line Parsing:



Typedefs

- typedef void(* [parse_cmd_fn_t](#)) (int)
Function type for PARSE_CMD_FN.
- typedef void(* [parse_cmd_fn_arg_t](#)) (int, const char *, int)
Function type for PARSE_CMD_FN_ARG.

Enumerations

- enum [parse_cmd_type](#)
Types for parsing.

Functions

- int [parse_cmdline](#) (int *argc, const char ***argv, char arg0,...)
Parse the command-line for specified arguments and store the values into variables.

12.15.1 Detailed Description

12.15.2 Function Documentation

12.15.2.1 `parse_cmdline()`

```
int parse_cmdline (
    int * argc,
    const char *** argv,
    char arg0,
    ... )
```

Parse the command-line for specified arguments and store the values into variables.

This Functions gets the command-line, and a list of command-descriptors. Then, the command-line is parsed according to the given descriptors, storing strings, switches and numeric arguments at given addresses, and possibly calling specified functions. A default help descriptor is added. Its purpose is to present a short command overview in the case the given command-line does not fit to the descriptors.

Each command-descriptor has the following form:

short option char, long option name, comment, type, val, addr.

The *short option char* specifies the short form of the described option. The short form will be recognized after a single dash, or in a group of short options preceded by a single dash. Specify ' ' if no short form should be used.

The *long option name* specifies the long form of the described option. The long form will be recognized after two dashes. Specify 0 if no long form should be used for this option.

The *comment* is a string that will be used when presenting the short command-line help.

The *type* specifies, if the option should be recognized as

- a number (`PARSE_CMD_INT`),
- a switch (`PARSE_CMD_SWITCH`),
- a string (`PARSE_CMD_STRING`),
- a function call (`PARSE_CMD_FN`, `PARSE_CMD_FN_ARG`),
- an increment/decrement operator (`PARSE_CMD_INC`, `PARSE_CMD_DEC`).

If *type* is `PARSE_CMD_INT`, the option requires a second argument on the command-line after the option. This argument is parsed as a number. It can be preceded by 0x to present a hex-value or by 0 to present an octal form. *addr* is interpreted as an int-pointer. The scanned argument from the command-line is stored in this pointer.

If *type* is `PARSE_CMD_SWITCH`, *addr* must be a pointer to int, and the value from *val* is stored at this pointer.

With `PARSE_CMD_STRING`, an additional argument is expected at the cmdline. *addr* must be a pointer to `const char*`, and a pointer to the argument on the command line is stored at this pointer. The value in *val* is a default value, which is stored at *addr* if the corresponding option is not given on the command line.

With `PARSE_CMD_FN_ARG`, *addr* is interpreted as a function pointer of type `parse_cmd_fn_t`. It will be called with *val* as argument if the corresponding option is found.

If *type* is `PARSE_CMD_FN_ARG`, *addr* is as a function pointer of type `parse_cmd_fn_arg_t`, and handled similar to `PARSE_CMD_FN`. An additional argument is expected at the command line, however. It is given to the called function as 2nd argument, and parsed as an integer as with `PARSE_CMD_INT` as a third argument.

If *type* is `PARSE_CMD_INC` or `PARSE_CMD_DEC`, *addr* is interpreted as an int-pointer. The value of *val* is stored to this pointer first. For every occurrence of the option in the command line, the integer referenced by *addr* is incremented or decremented, respectively.

The list of command-descriptors is terminated by specifying a binary 0 for the short option char.

Note: The short option char 'h' and the long option name "help" must not be specified. They are used for the default help descriptor and produce a short command-options help when specified on the command-line.

Parameters

<i>argc</i>	pointer to number of command line parameters as passed to main
<i>argv</i>	pointer to array of command line parameters as passed to main
<i>arg0</i>	format list describing the command line options to parse for

Returns

0 if the command-line was successfully parsed, otherwise:

- -1 if the given descriptors are somehow wrong.
- -2 if not enough memory was available to hold temporary structs.
- -3 if the given command-line args did not meet the specified set.
- -4 if the help-option was given.

Upon return, *argc* and *argv* point to a list of arguments that were not scanned as arguments. See `getoptlong` for details on scanning.

12.16 Console API

[Console](#) interface.

Collaboration diagram for Console API:



Data Structures

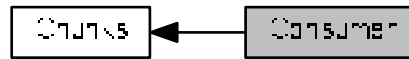
- class [L4Re::Console](#)
Console class.

12.16.1 Detailed Description

[Console](#) interface.

12.17 Consumer

Collaboration diagram for Consumer:



Functions

- long [l4shmc_enable_chunk](#) (l4shmc_chunk_t *chunk)
Enable a signal connected with a chunk.
- long [l4shmc_wait_chunk](#) (l4shmc_chunk_t *chunk)
Wait on a specific chunk.
- long [l4shmc_wait_chunk_to](#) (l4shmc_chunk_t *chunk, [l4_timeout_t](#) timeout)
Check whether a specific chunk has an event pending, with timeout.
- long [l4shmc_wait_chunk_try](#) (l4shmc_chunk_t *chunk)
Check whether a specific chunk has an event pending.
- long [l4shmc_chunk_consumed](#) (l4shmc_chunk_t *chunk)
Mark a chunk as free.
- long [l4shmc_is_chunk_ready](#) (l4shmc_chunk_t *chunk)
Check whether data is available.
- long [l4shmc_chunk_size](#) (l4shmc_chunk_t *chunk)
Get current size of a chunk.

12.17.1 Detailed Description

12.17.2 Function Documentation

12.17.2.1 l4shmc_chunk_consumed()

```
long l4shmc_chunk_consumed (
    l4shmc_chunk_t * chunk ) [inline]
```

Mark a chunk as free.

Parameters

<i>chunk</i>	Chunk to mark as free.
--------------	------------------------

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.17.2.2 l4shmc_chunk_size()

```
long l4shmc_chunk_size (
    l4shmc_chunk_t * chunk ) [inline]
```

Get current size of a chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.17.2.3 l4shmc_enable_chunk()

```
long l4shmc_enable_chunk (
    l4shmc_chunk_t * chunk )
```

Enable a signal connected with a chunk.

Parameters

<i>chunk</i>	Chunk to enable.
--------------	------------------

Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

12.17.2.4 l4shmc_is_chunk_ready()

```
long l4shmc_is_chunk_ready (
    l4shmc_chunk_t * chunk ) [inline]
```

Check whether data is available.

Parameters

<i>chunk</i>	Chunk to check.
--------------	-----------------

Return values

0	Success.
<0	Error.

12.17.2.5 l4shmc_wait_chunk()

```
long l4shmc_wait_chunk (
    l4shmc_chunk_t * chunk ) [inline]
```

Wait on a specific chunk.

Parameters

<i>chunk</i>	Chunk to wait for.
--------------	--------------------

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.17.2.6 l4shmc_wait_chunk_to()

```
long l4shmc_wait_chunk_to (
    l4shmc_chunk_t * chunk,
    l4_timeout_t timeout )
```

Check whether a specific chunk has an event pending, with timeout.

Parameters

<i>chunk</i>	Chunk to check.
<i>timeout</i>	Timeout.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

12.17.2.7 l4shmc_wait_chunk_try()

```
long l4shmc_wait_chunk_try (
    l4shmc_chunk_t * chunk ) [inline]
```

Check whether a specific chunk has an event pending.

Parameters

<i>chunk</i>	Chunk to check.
--------------	-----------------

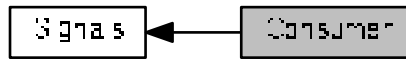
Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

12.18 Consumer

Collaboration diagram for Consumer:



Functions

- long [l4shmc_enable_signal](#) (l4shmc_signal_t *signal)
Enable a signal.
- long [l4shmc_wait_any](#) (l4shmc_signal_t **retsignal)
Wait on any signal.
- long [l4shmc_wait_any_try](#) (l4shmc_signal_t **retsignal)
Check whether any waited signal has an event pending.
- long [l4shmc_wait_any_to](#) (l4_timeout_t timeout, l4shmc_signal_t **retsignal)
Wait for any signal with timeout.
- long [l4shmc_wait_signal](#) (l4shmc_signal_t *signal)
Wait on a specific signal.
- long [l4shmc_wait_signal_to](#) (l4shmc_signal_t *signal, l4_timeout_t timeout)
Wait on a specific signal, with timeout.
- long [l4shmc_wait_signal_try](#) (l4shmc_signal_t *signal)
Check whether a specific signal has an event pending.

12.18.1 Detailed Description

12.18.2 Function Documentation

12.18.2.1 l4shmc_enable_signal()

```
long l4shmc_enable_signal (
    l4shmc_signal_t * signal )
```

Enable a signal.

Parameters

<i>signal</i>	Signal to enable.
---------------	-------------------

Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

12.18.2.2 l4shmc_wait_any()

```
long l4shmc_wait_any (
    l4shmc_signal_t ** retsignal ) [inline]
```

Wait on any signal.

Parameters

out	<i>retsignal</i>	Signal received.
-----	------------------	------------------

Return values

0	Success.
<0	Error.

12.18.2.3 l4shmc_wait_any_to()

```
long l4shmc_wait_any_to (
    l4_timeout_t timeout,
    l4shmc_signal_t ** retsignal )
```

Wait for any signal with timeout.

Parameters

	<i>timeout</i>	Timeout.
out	<i>retsignal</i>	Signal that has the event pending if any.

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

12.18.2.4 l4shmc_wait_any_try()

```
long l4shmc_wait_any_try (
    l4shmc_signal_t ** retsignal ) [inline]
```

Check whether any waited signal has an event pending.

Parameters

<i>out</i>	<i>retsignal</i>	Signal that has the event pending if any.
------------	------------------	---

Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

12.18.2.5 l4shmc_wait_signal()

```
long l4shmc_wait_signal (
    l4shmc_signal_t * signal ) [inline]
```

Wait on a specific signal.

Parameters

<i>signal</i>	Signal to wait for.
---------------	---------------------

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.18.2.6 l4shmc_wait_signal_to()

```
long l4shmc_wait_signal_to (
    l4shmc_signal_t * signal,
    l4_timeout_t timeout )
```

Wait on a specific signal, with timeout.

Parameters

<i>signal</i>	Signal to wait for.
<i>timeout</i>	Timeout.

Return values

<i>0</i>	Success.
<i><0</i>	Error.

12.18.2.7 l4shmc_wait_signal_try()

```
long l4shmc_wait_signal_try (
    l4shmc_signal_t * signal ) [inline]
```

Check whether a specific signal has an event pending.

Parameters

<i>signal</i>	Signal to check.
---------------	------------------

Return values

<i>0</i>	Success.
<i><0</i>	Error.

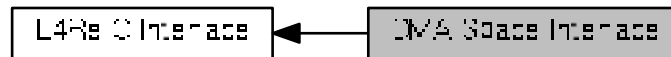
The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

12.19 DMA API for L4Re

12.20 DMA Space Interface

DMA Space C interface.

Collaboration diagram for DMA Space Interface:



Typedefs

- typedef [l4_cap_idx_t](#) [l4re_dma_space_t](#)
DMA space capability type.

Functions

- long [l4re_dma_space_map](#) ([l4re_dma_space_t](#) dma, [l4re_ds_t](#) src, [l4_addr_t](#) offset, [l4_size_t](#) *size, unsigned long attrs, enum [l4re_dma_space_direction](#) dir, [l4re_dma_space_dma_addr_t](#) *dma_addr) [L4_NOTHROW](#)
Map the given part of this data space into the DMA address space.
- long [l4re_dma_space_unmap](#) ([l4re_dma_space_t](#) dma, [l4re_dma_space_dma_addr_t](#) dma_addr, [l4_size_t](#) size, unsigned long attrs, enum [l4re_dma_space_direction](#) dir) [L4_NOTHROW](#)
Unmap the given part of this data space from the DMA address space.
- long [l4re_dma_space_associate](#) ([l4re_dma_space_t](#) dma, [l4_cap_idx_t](#) dma_task, unsigned long attr) [L4_NOTHROW](#)
Associate a DMA task for a device to this Dma_space.
- long [l4re_dma_space_disassociate](#) ([l4re_dma_space_t](#) dma)
Disassociate the DMA task from this Dma_space.

12.20.1 Detailed Description

DMA Space C interface.

12.20.2 Typedef Documentation

12.20.2.1 l4re_dma_space_t

```
typedef l4_cap_idx_t l4re_dma_space_t
```

DMA space capability type.

DMA Address Space. A `Dma_space` represents the DMA address space of a device. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a DMA address space, using `map()`, makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, `map()` is responsible for the necessary syncing operations before the DMA.

`unmap()` is the reverse operation to `map()` and unmaps the given data-space part for the DMA address space. `unmap()` is responsible for the necessary sync operations after the DMA.

Definition at line 59 of file [dma_space.h](#).

12.20.3 Function Documentation

12.20.3.1 l4re_dma_space_associate()

```
long l4re_dma_space_associate (
    l4re_dma_space_t dma,
    l4_cap_idx_t dma_task,
    unsigned long attr )
```

Associate a DMA task for a device to this `Dma_space`.

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {RW}

Parameters

in	<i>dma_task</i>	The DMA task used for the device that shall be associated with this DMA space. The <code>dma_task</code> might be an invalid capability when <code>#Phys_space</code> is set in <code>attr</code> , in this case the CPUs physical memory is used as DMA address space.
in	<i>attr</i>	Attributes for this DMA space. See <code>#Space_attrb</code> .

12.20.3.2 l4re_dma_space_disassociate()

```
long l4re_dma_space_disassociate (
    l4re_dma_space_t dma )
```

Disassociate the DMA task from this Dma_space.

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {RW}

12.20.3.3 l4re_dma_space_map()

```
long l4re_dma_space_map (
    l4re_dma_space_t dma,
    l4re_ds_t src,
    l4_addr_t offset,
    l4_size_t * size,
    unsigned long attrs,
    enum l4re_dma_space_direction dir,
    l4re_dma_space_dma_addr_t * dma_addr )
```

Map the given part of this data space into the DMA address space.

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {R}

Parameters

in	<i>src</i>	Source data space (that describes the memory). Caller needs write rights to the data space.
in	<i>offset</i>	The offset (bytes) within <i>src</i> .
in, out	<i>size</i>	The size (bytes) of the of the region to be mapped to for DMA, after successful mapping the size returned is the size mapped for DMA as single block, this size might be smaller than the original input size, in this case the caller might call map() again with a new offset and the remaining size.
in	<i>attrs</i>	The attributes used for this DMA mapping (a combination of Dma_space::Attribute values).
in	<i>dir</i>	The direction of the DMA transfer issued with this mapping. The same value must later be passed to unmap().
out	<i>dma_addr</i>	The DMA address to use for DMA with the associated device.

Returns

0 in the case of success, a negative error code otherwise.

12.20.3.4 l4re_dma_space_unmap()

```
long l4re_dma_space_unmap (
    l4re_dma_space_t dma,
    l4re_dma_space_dma_addr_t dma_addr,
    l4_size_t size,
    unsigned long attrs,
    enum l4re_dma_space_direction dir )
```

Unmap the given part of this data space from the DMA address space.

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {R}

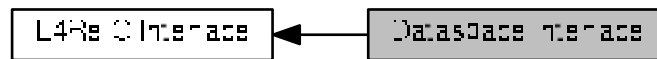
Parameters

<i>dma_addr</i>	The DMA address (returned by <code>Dma_space::map()</code>).
<i>size</i>	The size (bytes) of the memory region to unmap.
<i>attrs</i>	The attributes for the unmap (currently none).
<i>dir</i>	The direction of the finished DMA operation.

12.21 Dataspace interface

Dataspace C interface.

Collaboration diagram for Dataspace interface:



Data Structures

- struct [l4re_ds_stats_t](#)
Information about the data space.

Typedefs

- typedef [l4_cap_idx_t](#) [l4re_ds_t](#)
Dataspace type.
- typedef [l4_cap_idx_t](#) [l4re_namespace_t](#)
Namespace type.

Enumerations

- enum [l4re_ds_map_flags](#) { ,
[L4RE_DS_MAP_NORMAL](#) = 0x00, [L4RE_DS_MAP_CACHEABLE](#) = [L4RE_DS_MAP_NORMAL](#), [L4RE_DS_MAP_BUFFERABLE](#) = 0x10, [L4RE_DS_MAP_UNCACHEABLE](#) = 0x20,
[L4RE_DS_MAP_CACHING_MASK](#) = 0x30, [L4RE_DS_MAP_CACHING_SHIFT](#) = 4 }
- Flags to specify the memory mapping type of a request.*

Functions

- long [l4re_ds_clear](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) offset, unsigned long size) [L4_NOTHROW](#)
- long [l4re_ds_allocate](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) offset, [l4_size_t](#) size) [L4_NOTHROW](#)
- int [l4re_ds_copy_in](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) dst_offs, const [l4re_ds_t](#) src, [l4_addr_t](#) src_offs, unsigned long size) [L4_NOTHROW](#)
- unsigned long [l4re_ds_size](#) (const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
- long [l4re_ds_flags](#) (const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
- int [l4re_ds_info](#) (const [l4re_ds_t](#) ds, [l4re_ds_stats_t](#) *stats) [L4_NOTHROW](#)
- int [l4re_ds_phys](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) offset, [l4_addr_t](#) *phys_addr, [l4_size_t](#) *phys_size) [L4_NOTHROW](#)
Return physical address.

12.21.1 Detailed Description

Dataspace C interface.

12.21.2 Enumeration Type Documentation

12.21.2.1 l4re_ds_map_flags

```
enum l4re_ds_map_flags
```

Flags to specify the memory mapping type of a request.

Enumerator

L4RE_DS_MAP_NORMAL	request normal memory mapping
L4RE_DS_MAP_CACHEABLE	request normal memory mapping
L4RE_DS_MAP_BUFFERABLE	request bufferable (write buffered) mappings
L4RE_DS_MAP_UNCACHEABLE	request uncacheable memory mappings
L4RE_DS_MAP_CACHING_MASK	mask for caching flags
L4RE_DS_MAP_CACHING_SHIFT	shift value for caching flags

Definition at line 54 of file [dataspace.h](#).

12.21.3 Function Documentation

12.21.3.1 l4re_ds_allocate()

```
long l4re_ds_allocate (
    const l4re_ds_t ds,
    l4_addr_t offset,
    l4_size_t size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::allocate](#)

12.21.3.2 l4re_ds_clear()

```
long l4re_ds_clear (
    const l4re_ds_t ds,
    l4_addr_t offset,
    unsigned long size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::clear](#)

12.21.3.3 l4re_ds_copy_in()

```
int l4re_ds_copy_in (
    const l4re_ds_t ds,
    l4_addr_t dst_offs,
    const l4re_ds_t src,
    l4_addr_t src_offs,
    unsigned long size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::copy_in](#)

12.21.3.4 l4re_ds_flags()

```
long l4re_ds_flags (
    const l4re_ds_t ds )
```

See also

[L4Re::Dataspace::flags](#)

12.21.3.5 l4re_ds_info()

```
int l4re_ds_info (
    const l4re_ds_t ds,
    l4re_ds_stats_t * stats )
```

See also

[L4Re::Dataspace::info](#)

12.21.3.6 l4re_ds_phys()

```
int l4re_ds_phys (
    const l4re_ds_t ds,
    l4_addr_t offset,
    l4_addr_t * phys_addr,
    l4_size_t * phys_size )
```

Return physical address.

Parameters

<i>ds</i>	Dataspace
<i>offset</i>	Offset in bytes in dataspace

Return values

<i>phys_addr</i>	Physical address
<i>phys_size</i>	Size of physically contiguous region starting from <i>phys_addr</i> (in bytes).

Returns

0 for success, <0 on error

The function returns the physical address of an offset in a dataspace. Use multiple calls of the function to get all physical regions in case of physically non-contiguous dataspace.

See also

[L4Re::Dataspace::phys](#)

12.21.3.7 l4re_ds_size()

```
unsigned long l4re_ds_size (
    const l4re_ds_t ds )
```

Returns

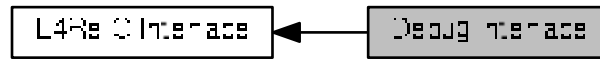
Size of the dataspace in bytes.

See also

[L4Re::Dataspace::size](#)

12.22 Debug interface

Collaboration diagram for Debug interface:



Functions

- long [l4re_debug_obj_debug](#) ([l4_cap_idx_t](#) *srv*, unsigned long *function*) [L4_NOTHROW](#)
Call debug function of [L4Re](#) service.

12.22.1 Detailed Description

12.22.2 Function Documentation

12.22.2.1 l4re_debug_obj_debug()

```
long l4re_debug_obj_debug (
    l4\_cap\_idx\_t srv,
    unsigned long function )
```

Call debug function of [L4Re](#) service.

Parameters

<i>srv</i>	Object to call.
<i>function</i>	Function to call.

See also

[L4Re::Debug_obj::debug](#)

12.23 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



Data Structures

- class [L4Re::Debug_obj](#)
Debug interface.

12.23.1 Detailed Description

Debugging Interface.

The debugging interface can be provided to retrieve, or log debugging information for an object. Each class may realize the debug interface to provide debugging functionality. For example, the region-map objects provide a facility to dump the currently established memory regions.

See also

[L4::Debug_obj](#) for more information.

12.24 EDID parsing functionality

Enumerations

- enum [Libedid_consts](#) { [Libedid_block_size](#) = 128 }
EDID constants.

Functions

- int [libedid_check_header](#) (const unsigned char *edid)
Check for valid EDID header.
- int [libedid_checksum](#) (const unsigned char *edid)
Calculates the EDID checksum.
- unsigned [libedid_version](#) (const unsigned char *edid)
Returns the EDID version number.
- unsigned [libedid_revision](#) (const unsigned char *edid)
Returns the EDID revision number.
- void [libedid_pnp_id](#) (const unsigned char *edid, unsigned char *id)
Extracts the display's PnP ID.
- void [libedid_preferred_resolution](#) (const unsigned char *edid, unsigned *w, unsigned *h)
Extract the display's preferred mode.
- unsigned [libedid_num_ext_blocks](#) (const unsigned char *edid)
Get the number of EDID extension blocks.
- unsigned [libedid_dump_standard_timings](#) (const unsigned char *edid)
Dump the standard timings to stdout.
- void [libedid_dump](#) (const unsigned char *edid)
Dump raw EDID data to stdout.

12.24.1 Detailed Description

12.24.2 Enumeration Type Documentation

12.24.2.1 Libedid_consts

enum [Libedid_consts](#)

EDID constants.

Enumerator

Libedid_block_size	Size of one EDID block in bytes.
------------------------------------	----------------------------------

Definition at line 23 of file [edid.h](#).

12.24.3 Function Documentation

12.24.3.1 libedid_check_header()

```
int libedid_check_header (
    const unsigned char * edid )
```

Check for valid EDID header.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

Returns

0 if the header is correct, -EINVAL otherwise

12.24.3.2 libedid_checksum()

```
int libedid_checksum (
    const unsigned char * edid )
```

Calculates the EDID checksum.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

Returns

0 if checksum is correct, -EINVAL otherwise

12.24.3.3 libedid_dump()

```
void libedid_dump (
    const unsigned char * edid )
```

Dump raw EDID data to stdout.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

12.24.3.4 libedid_dump_standard_timings()

```
unsigned libedid_dump_standard_timings (
    const unsigned char * edid )
```

Dump the standard timings to stdout.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

Returns

Number of standard timings stored in EDID

12.24.3.5 libedid_num_ext_blocks()

```
unsigned libedid_num_ext_blocks (
    const unsigned char * edid )
```

Get the number of EDID extension blocks.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

Returns

Number of EDID extension blocks

12.24.3.6 libedid_pnp_id()

```
void libedid_pnp_id (
    const unsigned char * edid,
    unsigned char * id )
```

Extracts the display's PnP ID.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

Return values

<i>id</i>	Return the PnP id. Must point to 4 bytes.
-----------	---

12.24.3.7 libedid_prefered_resolution()

```
void libedid_prefered_resolution (
    const unsigned char * edid,
    unsigned * w,
    unsigned * h )
```

Extract the display's preferred mode.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

Return values

<i>w</i>	X resolution of preferred video mode in pixels.
<i>h</i>	Y resolution of preferred video mode in pixels.

12.24.3.8 libedid_revision()

```
unsigned libedid_revision (
    const unsigned char * edid )
```

Returns the EDID revision number.

Parameters

<i>edid</i>	Pointer to a 128 EDID block
-------------	-----------------------------

Returns

Revision number

12.24.3.9 libedid_version()

```
unsigned libedid_version (
    const unsigned char * edid )
```

Returns the EDID version number.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

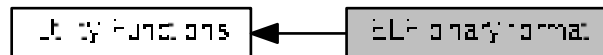
Returns

Version number

12.25 ELF binary format

Functions and types related to ELF binaries.

Collaboration diagram for ELF binary format:



Files

- file [elf.h](#)
ELF definition.

Data Structures

- struct [Elf32_Ehdr](#)
ELF32 header.
- struct [Elf64_Ehdr](#)
ELF64 header.
- struct [Elf32_Shdr](#)
ELF32 section header - figure 1-9, page 1-9.
- struct [Elf64_Shdr](#)
ELF64 section header.
- struct [Elf32_Phdr](#)
ELF32 program header.
- struct [Elf64_Phdr](#)
ELF64 program header.
- struct [Elf32_Dyn](#)
ELF32 dynamic entry.
- struct [Elf64_Dyn](#)
ELF64 dynamic entry.
- struct [Elf32_Sym](#)
ELF32 symbol table entry.
- struct [Elf64_Sym](#)
ELF64 symbol table entry.

Macros

- #define `EI_NIDENT` 16
number of characters
- #define `EI_CLASS` 4
ELF class byte index.
- #define `EI_CLASS` 4
ELF class byte index.
- #define `ELFCLASSNONE` 0
Invalid ELF class.
- #define `ELFCLASSNONE` 0
Invalid ELF class.
- #define `ELFCLASS32` 1
32-bit objects
- #define `ELFCLASS64` 2
64-bit objects
- #define `ELFCLASSNUM` 3
Mask for 32-bit or 64-bit class.
- #define `EI_DATA` 5
Data encoding byte index.
- #define `EI_DATA` 5
Data encoding byte index.
- #define `ELFDATANONE` 0
Invalid data encoding.
- #define `ELFDATANONE` 0
Invalid data encoding.
- #define `ELFDATA2LSB` 1
2's complement, little endian
- #define `ELFDATA2LSB` 1
2's complement, little endian
- #define `ELFDATA2MSB` 2
2's complement, big endian
- #define `ELFDATA2MSB` 2
2's complement, big endian
- #define `EI_VERSION` 6
File version byte index.
- #define `EI_VERSION` 6
File version byte index.
- #define `EI_OSABI` 7
OS ABI identification.
- #define `EI_OSABI` 7
OS ABI identification.
- #define `ELFOSABI_NONE` 0
UNIX System V ABI.
- #define `ELFOSABI_SYSV` 0
Alias.
- #define `ELFOSABI_SYSV` 0
Alias.
- #define `ELFOSABI_HPUX` 1
HP-UX.
- #define `ELFOSABI_HPUX` 1

- *HP-UX.*
- #define [ELFOSABI_NETBSD](#) 2
NetBSD.
- #define [ELFOSABI_LINUX](#) 3
Linux.
- #define [ELFOSABI_SOLARIS](#) 6
Sun Solaris.
- #define [ELFOSABI_AIX](#) 7
IBM AIX.
- #define [ELFOSABI_IRIX](#) 8
SGI Irix.
- #define [ELFOSABI_FREEBSD](#) 9
FreeBSD.
- #define [ELFOSABI_TRU64](#) 10
Compaq TRU64 UNIX.
- #define [ELFOSABI_MODESTO](#) 11
Novell Modesto.
- #define [ELFOSABI_OPENBSD](#) 12
OpenBSD.
- #define [ELFOSABI_ARM](#) 97
ARM.
- #define [ELFOSABI_STANDALONE](#) 255
Standalone (embedded) application.
- #define [ELFOSABI_STANDALONE](#) 255
Standalone (embedded) application.
- #define [EI_ABIVERSION](#) 8
ABI version.
- #define [EI_ABIVERSION](#) 8
ABI version.
- #define [EI_PAD](#) 9
Byte index of padding bytes.
- #define [EI_PAD](#) 9
Byte index of padding bytes.
- #define [ET_NONE](#) 0
no file type
- #define [ET_REL](#) 1
relocatable file
- #define [ET_EXEC](#) 2
executable file
- #define [ET_DYN](#) 3
shared object file
- #define [ET_CORE](#) 4
core file
- #define [ET_LOPROC](#) 0xff00
processor-specific
- #define [ET_HIPROC](#) 0xffff
processor-specific
- #define [EM_NONE](#) 0
no machine
- #define [EM_M32](#) 1
AT&T WE 32100.

- #define [EM_SPARC](#) 2
SPARC.
- #define [EM_386](#) 3
Intel 80386.
- #define [EM_68K](#) 4
Motorola 68000.
- #define [EM_88K](#) 5
Motorola 88000.
- #define [EM_860](#) 7
Intel 80860.
- #define [EM_MIPS](#) 8
MIPS RS3000 big-endian.
- #define [EM_MIPS_RS4_BE](#) 10
MIPS RS4000 big-endian.
- #define [EM_SPARC64](#) 11
SPARC 64-bit.
- #define [EM_PARISC](#) 15
HP PA-RISC.
- #define [EM_VPP500](#) 17
Fujitsu VPP500.
- #define [EM_SPARC32PLUS](#) 18
Sun's V8plus.
- #define [EM_960](#) 19
Intel 80960.
- #define [EM_PPC](#) 20
PowerPC.
- #define [EM_V800](#) 36
NEC V800.
- #define [EM_FR20](#) 37
Fujitsu FR20.
- #define [EM_RH32](#) 38
TRW RH-32.
- #define [EM_RCE](#) 39
Motorola RCE.
- #define [EM_ARM](#) 40
Advanced RISC Machines ARM.
- #define [EM_ALPHA](#) 41
Digital Alpha.
- #define [EM_SH](#) 42
Hitachi SuperH.
- #define [EM_SPARCV9](#) 43
SPARC v9 64-bit.
- #define [EM_TRICORE](#) 44
Siemens Tricore embedded processor.
- #define [EM_ARC](#) 45
Argonaut RISC Core, Argonaut Techn Inc.
- #define [EM_H8_300](#) 46
Hitachi H8/300.
- #define [EM_H8_300H](#) 47
Hitachi H8/300H.
- #define [EM_H8S](#) 48

- Hitachi H8/S.
 • #define [EM_H8_500](#) 49
- Hitachi H8/500.
 • #define [EM_IA_64](#) 50
- HP/Intel IA-64.
 • #define [EM_MIPS_X](#) 51
- Stanford MIPS-X.
 • #define [EM_COLDFIRE](#) 52
- Motorola Coldfire.
 • #define [EM_68HC12](#) 53
- Motorola M68HC12.
 • #define [EM_X86_64](#) 62
- Advanced Micro Devices x86-64.
 • #define [EM_PDSP](#) 63
- Sony DSP Processor.
 • #define [EM_FX66](#) 66
- Siemens FX66 microcontroller.
 • #define [EM_ST9PLUS](#) 67
- STMicroelectronics ST9+ 8/16 mc.
 • #define [EM_ST7](#) 68
- STmicroelectronics ST7 8 bit mc.
 • #define [EM_68HC16](#) 69
- Motorola MC68HC16 microcontroller.
 • #define [EM_68HC11](#) 70
- Motorola MC68HC11 microcontroller.
 • #define [EM_68HC08](#) 71
- Motorola MC68HC08 microcontroller.
 • #define [EM_68HC05](#) 72
- Motorola MC68HC05 microcontroller.
 • #define [EM_SVX](#) 73
- Silicon Graphics SVx.
 • #define [EM_ST19](#) 74
- STMicroelectronics ST19 8 bit mc.
 • #define [EM_VAX](#) 75
- Digital VAX.
 • #define [EM_CRIS](#) 76
- Axis Communications 32-bit embedded processor.
 • #define [EM_JAVELIN](#) 77
- Infineon Technologies 32-bit embedded processor.
 • #define [EM_FIREPATH](#) 78
- Element 14 64-bit DSP Processor.
 • #define [EM_ZSP](#) 79
- LSI Logic 16-bit DSP Processor.
 • #define [EM_MMIX](#) 80
- Donald Knuth's educational 64-bit processor.
 • #define [EM_HUANY](#) 81
- Harvard University machine-independent object files.
 • #define [EM_PRISM](#) 82
- SiTera Prism.
 • #define [EM_AVR](#) 83
- Atmel AVR 8-bit microcontroller.

- #define [EM_FR30](#) 84
Fujitsu FR30.
- #define [EM_D10V](#) 85
Mitsubishi D10V.
- #define [EM_D30V](#) 86
Mitsubishi D30V.
- #define [EM_V850](#) 87
NEC v850.
- #define [EM_M32R](#) 88
Mitsubishi M32R.
- #define [EM_MN10300](#) 89
Matsushita MN10300.
- #define [EM_MN10200](#) 90
Matsushita MN10200.
- #define [EM_PJ](#) 91
picoJava
- #define [EM_OPENRISC](#) 92
OpenRISC 32-bit embedded processor.
- #define [EM_ARC_A5](#) 93
ARC Cores Tangent-A5.
- #define [EM_XTENSA](#) 94
Tensilica Xtensa Architecture.
- #define [EM_ALTERA_NIOS2](#) 113
Altera Nios II.
- #define [EM_AARCH64](#) 183
ARM AARCH64.
- #define [EM_TILEPRO](#) 188
Tilera TILEPro.
- #define [EM_MICROBLAZE](#) 189
Xilinx MicroBlaze.
- #define [EM_TILEGX](#) 191
Tilera TILE-Gx.
- #define [EV_NONE](#) 0
Invalid version.
- #define [EV_CURRENT](#) 1
Current version.
- #define [EI_MAG0](#) 0
file id
- #define [EI_MAG1](#) 1
file id
- #define [EI_MAG2](#) 2
file id
- #define [EI_MAG3](#) 3
file id
- #define [ELFMAG0](#) 0x7f
e_ident[EI_MAG0]
- #define [ELFMAG1](#) 'E'
e_ident[EI_MAG1]
- #define [ELFMAG2](#) 'L'
e_ident[EI_MAG2]
- #define [ELFMAG3](#) 'F'

- `e_ident[EI_MAG3]`
- `#define ELFCLASS32 1`
32-bit object
- `#define ELFCLASS64 2`
64-bit object
- `#define SHN_UNDEF 0`
undefined section header entry
- `#define SHN_LORESERVE 0xff00`
lower bound of reserved indexes
- `#define SHN_LOPROC 0xff00`
lower bound of proc spec entr
- `#define SHN_HIPROC 0xff1f`
upper bound of proc spec entr
- `#define SHN_ABS 0xffff`
absolute values for ref
- `#define SHN_COMMON 0xffff2`
common symbols
- `#define SHN_HIRESERVE 0xffff`
upper bound of reserved indexes
- `#define SHT_INIT_ARRAY 14`
Array of constructors.
- `#define SHT_FINI_ARRAY 15`
Array of destructors.
- `#define SHT_PREINIT_ARRAY 16`
Array of pre-constructors.
- `#define SHT_GROUP 17`
Section group.
- `#define SHT_SYMTAB_SHNDX 18`
Extended section indeces.
- `#define SHT_NUM 19`
Number of defined types.
- `#define SHF_WRITE 0x1`
writable during execution
- `#define SHF_ALLOC 0x2`
section occupies virt memory
- `#define SHF_EXECINSTR 0x4`
code section
- `#define SHF_MERGE 0x10`
Might be merged.
- `#define SHF_STRINGS 0x20`
Contains nul-terminated strings.
- `#define SHF_INFO_LINK 0x40`
'sh_info' contains SHT index
- `#define SHF_LINK_ORDER 0x80`
Preserve order after combining.
- `#define SHF_OS_NONCONFORMING 0x100`
Non-standard OS specific handling required.
- `#define SHF_GROUP 0x200`
Section is member of a group.
- `#define SHF_TLS 0x400`
Section hold thread-local data.

- #define SHF_MASKOS 0x0ff00000
OS-specific.
- #define SHF_MASKPROC 0xf0000000
proc spec mask
- #define PT_NULL 0
array is unused
- #define PT_LOAD 1
loadable
- #define PT_DYNAMIC 2
dynamic linking information
- #define PT_INTERP 3
path to interpreter
- #define PT_NOTE 4
auxiliary information
- #define PT_SHLIB 5
reserved
- #define PT_PHDR 6
location of the pht itself
- #define PT_TLS 7
Thread-local storage segment.
- #define PT_NUM 8
Number of defined types.
- #define PT_LOOS 0x60000000
os spec.
- #define PT_HIOS 0x6fffffff
os spec.
- #define PT_LOPROC 0x70000000
processor spec.
- #define PT_HIPROC 0x7fffffff
processor spec.
- #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
EH frame information.
- #define PT_GNU_STACK (PT_LOOS + 0x474e551)
Flags for stack.
- #define PT_GNU_RELRO (PT_LOOS + 0x474e552)
Read only after reloc.
- #define PT_L4_STACK (PT_LOOS + 0x12)
Address of the stack.
- #define PT_L4_KIP (PT_LOOS + 0x13)
Address of the KIP.
- #define PT_L4_AUX (PT_LOOS + 0x14)
Address of the AUX strcutures.
- #define NT_PRSTATUS 1
Contains copy of prstatus struct.
- #define NT_FPREGSET 2
Contains copy of fpregset struct.
- #define NT_PRPSINFO 3
Contains copy of prpsinfo struct.
- #define NT_PRXREG 4
Contains copy of prxregset struct.
- #define NT_TASKSTRUCT 4

- Contains copy of task structure.*
- #define [NT_PLATFORM](#) 5
 - String from sysinfo(SI_PLATFORM)*
- #define [NT_AUXV](#) 6
 - Contains copy of auxv array.*
- #define [NT_GWINDOWS](#) 7
 - Contains copy of gwindows struct.*
- #define [NT_ASRS](#) 8
 - Contains copy of asrset struct.*
- #define [NT_PSTATUS](#) 10
 - Contains copy of pstatus struct.*
- #define [NT_PSINFO](#) 13
 - Contains copy of psinfo struct.*
- #define [NT_PRCRED](#) 14
 - Contains copy of prcred struct.*
- #define [NT_UTSNAME](#) 15
 - Contains copy of utsname struct.*
- #define [NT_LWPSTATUS](#) 16
 - Contains copy of lwpstatus struct.*
- #define [NT_LWPSINFO](#) 17
 - Contains copy of lwpinfo struct.*
- #define [NT_PRFPXREG](#) 20
 - Contains copy of fprxregset struct.*
- #define [NT_VERSION](#) 1
 - Contains a version string.*
- #define [DT_NULL](#) 0
 - Dynamic Array Tags, d_tag - figure 2-10, page 2-12.*
- #define [DT_NEEDED](#) 1
 - name of a needed library*
- #define [DT_PLTRELSZ](#) 2
 - total size of relocation entry*
- #define [DT_PLTGOT](#) 3
 - address assoc with prog link table*
- #define [DT_HASH](#) 4
 - address of symbol hash table*
- #define [DT_STRTAB](#) 5
 - address of string table*
- #define [DT_SYMTAB](#) 6
 - address of symbol table*
- #define [DT_RELA](#) 7
 - address of relocation table*
- #define [DT_RELASZ](#) 8
 - total size of relocation table*
- #define [DT_RELAENT](#) 9
 - size of DT_RELA relocation entry*
- #define [DT_STRSZ](#) 10
 - size of the string table*
- #define [DT_SYMENT](#) 11
 - size of a symbol table entry*
- #define [DT_INIT](#) 12
 - address of initialization function*

- #define [DT_FINI](#) 13
address of termination function
- #define [DT_SONAME](#) 14
name of the shared object
- #define [DT_RPATH](#) 15
search library path
- #define [DT_SYMBOLIC](#) 16
alter symbol resolution algorithm
- #define [DT_REL](#) 17
address of relocation table
- #define [DT_RELSZ](#) 18
total size of DT_REL relocation table
- #define [DT_RELENT](#) 19
size of the DT_REL relocation entry
- #define [DT_PTRREL](#) 20
type of relocation entry
- #define [DT_DEBUG](#) 21
for debugging purposes
- #define [DT_TEXTREL](#) 22
at least on entry changes r/o section
- #define [DT_JMPREL](#) 23
address of relocation entries
- #define [DT_BIND_NOW](#) 24
Process relocations of object.
- #define [DT_INIT_ARRAY](#) 25
Array with addresses of init fct.
- #define [DT_FINI_ARRAY](#) 26
Array with addresses of fini fct.
- #define [DT_INIT_ARRAYSZ](#) 27
Size in bytes of DT_INIT_ARRAY.
- #define [DT_FINI_ARRAYSZ](#) 28
Size in bytes of DT_FINI_ARRAY.
- #define [DT_RUNPATH](#) 29
Library search path.
- #define [DT_FLAGS](#) 30
Flags for the object being loaded.
- #define [DT_ENCODING](#) 32
Start of encoded range.
- #define [DT_PREINIT_ARRAY](#) 32
Array with addresses of preinit fct.
- #define [DT_PREINIT_ARRAYSZ](#) 33
size in bytes of DT_PREINIT_ARRAY
- #define [DT_NUM](#) 34
Number used.
- #define [DT_LOOS](#) 0x6000000d
Start of OS-specific.
- #define [DT_HIOS](#) 0x6ffff000
End of OS-specific.
- #define [DT_LOPROC](#) 0x70000000
processor spec.
- #define [DT_HIPROC](#) 0x7fffffff

- processor spec.*

 - #define `DF_ORIGIN` 0x00000001

Object may use DF_ORIGIN.
 - #define `DF_SYMBOLIC` 0x00000002

Symbol resolutions starts here.
 - #define `DF_TEXTREL` 0x00000004

Object contains text relocations.
 - #define `DF_BIND_NOW` 0x00000008

No lazy binding for this object.
 - #define `DF_STATIC_TLS` 0x00000010

Module uses the static TLS model.
 - #define `DF_1_NOW` 0x00000001

Set RTLD_NOW for this object.
 - #define `DF_1_GLOBAL` 0x00000002

Set RTLD_GLOBAL for this object.
 - #define `DF_1_GROUP` 0x00000004

Set RTLD_GROUP for this object.
 - #define `DF_1_NODELETE` 0x00000008

Set RTLD_NODELETE for this object.
 - #define `DF_1_LOADFLTR` 0x00000010

Trigger filtee loading at runtime.
 - #define `DF_1_INITFIRST` 0x00000020

Set RTLD_INITFIRST for this object.
 - #define `DF_1_NOOPEN` 0x00000040

Set RTLD_NOOPEN for this object.
 - #define `DF_1_ORIGIN` 0x00000080

\$ORIGIN must be handled.
 - #define `DF_1_DIRECT` 0x00000100

Direct binding enabled.
 - #define `DF_1_INTERPOSE` 0x00000400

Object is used to interpose.
 - #define `DF_1_NODEFLIB` 0x00000800

Ignore default lib search path.
 - #define `DF_1_NODUMP` 0x00001000

Object can't be dldump'ed.
 - #define `DF_1_CONFALT` 0x00002000

Configuration alternative created.
 - #define `DF_1_ENDFILTEE` 0x00004000

Filtee terminates filters search.
 - #define `DF_1_DISPRELDNE` 0x00008000

Disp reloc applied at build time.
 - #define `DF_1_DISPRELPND` 0x00010000

Disp reloc applied at run-time.
 - #define `DF_P1_LAZYLOAD` 0x00000001

Lazyload following object.
 - #define `DF_P1_GROUPPERM` 0x00000002

Symbols from next object are not generally available.
 - #define `R_386_NONE` 0

none
 - #define `R_386_32` 1

S + A.

- #define [R_386_PC32](#) 2
S + A - P
- #define [R_386_GOT32](#) 3
G + A - P
- #define [R_386_PLT32](#) 4
L + A - P
- #define [R_386_COPY](#) 5
none
- #define [R_386_GLOB_DAT](#) 6
S
- #define [R_386_JMP_SLOT](#) 7
S
- #define [R_386_RELATIVE](#) 8
B + A
- #define [R_386_GOTOFF](#) 9
S + A - GOT
- #define [R_386_GOTPC](#) 10
GOT + A - P
- #define [STB_LOCAL](#) 0
not visible outside object file
- #define [STB_GLOBAL](#) 1
visible to all objects beeing combined
- #define [STB_WEAK](#) 2
resemble global symbols
- #define [STB_LOOS](#) 10
os specific
- #define [STB_HIOS](#) 12
os specific
- #define [STB_LOPROC](#) 13
proc specific
- #define [STB_HIPROC](#) 15
proc specific
- #define [STT_NOTYPE](#) 0
symbol's type not specified
- #define [STT_OBJECT](#) 1
associated with a data object
- #define [STT_FUNC](#) 2
associated with a function or other code
- #define [STT_SECTION](#) 3
associated with a section
- #define [STT_FILE](#) 4
source file name associated with object
- #define [STT_LOOS](#) 10
os specific
- #define [STT_HIOS](#) 12
os specific
- #define [STT_LOPROC](#) 13
proc specific
- #define [STT_HIPROC](#) 15
proc specific

ELF types

- typedef [l4_uint32_t](#) Elf32_Addr
size 4 align 4
- typedef [l4_uint32_t](#) Elf32_Off
size 4 align 4
- typedef [l4_uint16_t](#) Elf32_Half
size 2 align 2
- typedef [l4_uint32_t](#) Elf32_Word
size 4 align 4
- typedef [l4_int32_t](#) Elf32_Sword
size 4 align 4
- typedef [l4_uint64_t](#) Elf64_Addr
size 8 align 8
- typedef [l4_uint64_t](#) Elf64_Off
size 8 align 8
- typedef [l4_uint16_t](#) Elf64_Half
size 2 align 2
- typedef [l4_uint32_t](#) Elf64_Word
size 4 align 4
- typedef [l4_int32_t](#) Elf64_Sword
size 4 align 4
- typedef [l4_uint64_t](#) Elf64_Xword
size 8 align 8
- typedef [l4_int64_t](#) Elf64_Sxword
size 8 align 8

12.25.1 Detailed Description

Functions and types related to ELF binaries.

12.25.2 Macro Definition Documentation

12.25.2.1 DF_1_CONFALT

```
#define DF_1_CONFALT 0x00002000
```

Configuration alternative created.

Definition at line [581](#) of file [elf.h](#).

12.25.2.2 DF_1_DIRECT

```
#define DF_1_DIRECT 0x00000100
```

Direct binding enabled.

Definition at line 576 of file [elf.h](#).

12.25.2.3 DF_1_DISPRELDNE

```
#define DF_1_DISPRELDNE 0x00008000
```

Disp reloc applied at build time.

Definition at line 583 of file [elf.h](#).

12.25.2.4 DF_1_DISPRELPND

```
#define DF_1_DISPRELPND 0x00010000
```

Disp reloc applied at run-time.

Definition at line 584 of file [elf.h](#).

12.25.2.5 DF_1_ENDFILTEE

```
#define DF_1_ENDFILTEE 0x00004000
```

Filtee terminates filters search.

Definition at line 582 of file [elf.h](#).

12.25.2.6 DF_1_GLOBAL

```
#define DF_1_GLOBAL 0x00000002
```

Set RTLD_GLOBAL for this object.

Definition at line 569 of file [elf.h](#).

12.25.2.7 DF_1_GROUP

```
#define DF_1_GROUP 0x00000004
```

Set RTLD_GROUP for this object.

Definition at line 570 of file [elf.h](#).

12.25.2.8 DF_1_INTERPOSE

```
#define DF_1_INTERPOSE 0x00000400
```

Object is used to interpose.

Definition at line 578 of file [elf.h](#).

12.25.2.9 DF_1_LOADFLTR

```
#define DF_1_LOADFLTR 0x00000010
```

Trigger filtee loading at runtime.

Definition at line 572 of file [elf.h](#).

12.25.2.10 DF_1_NODEFLIB

```
#define DF_1_NODEFLIB 0x00000800
```

Ignore default lib search path.

Definition at line 579 of file [elf.h](#).

12.25.2.11 DF_1_NODELETE

```
#define DF_1_NODELETE 0x00000008
```

Set RTLD_NODELETE for this object.

Definition at line 571 of file [elf.h](#).

12.25.2.12 DF_1_NODUMP

```
#define DF_1_NODUMP 0x00001000
```

Object can't be dldump'ed.

Definition at line [580](#) of file [elf.h](#).

12.25.2.13 DF_1_NOOPEN

```
#define DF_1_NOOPEN 0x00000040
```

Set RTLD_NOOPEN for this object.

Definition at line [574](#) of file [elf.h](#).

12.25.2.14 DF_1_NOW

```
#define DF_1_NOW 0x00000001
```

Set RTLD_NOW for this object.

Definition at line [568](#) of file [elf.h](#).

12.25.2.15 DF_1_ORIGIN

```
#define DF_1_ORIGIN 0x00000080
```

\$ORIGIN must be handled.

Definition at line [575](#) of file [elf.h](#).

12.25.2.16 DF_P1_GROUPPERM

```
#define DF_P1_GROUPPERM 0x00000002
```

Symbols from next object are not generally available.

Definition at line [592](#) of file [elf.h](#).

12.25.2.17 DF_P1_LAZYLOAD

```
#define DF_P1_LAZYLOAD 0x00000001
```

Lazyload following object.

Definition at line 591 of file [elf.h](#).

12.25.2.18 DT_HIPROC

```
#define DT_HIPROC 0x7fffffff
```

processor spec.

Definition at line 557 of file [elf.h](#).

12.25.2.19 DT_LOPROC

```
#define DT_LOPROC 0x70000000
```

processor spec.

Definition at line 556 of file [elf.h](#).

12.25.2.20 DT_NULL

```
#define DT_NULL 0
```

Dynamic Array Tags, d_tag - figure 2-10, page 2-12.

end of _DYNAMIC array

Definition at line 519 of file [elf.h](#).

12.25.2.21 EI_CLASS [1/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file [elf.h](#).

12.25.2.22 EI_CLASS [2/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file [elf.h](#).

12.25.2.23 EI_DATA [1/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file [elf.h](#).

12.25.2.24 EI_DATA [2/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file [elf.h](#).

12.25.2.25 EI_OSABI [1/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file [elf.h](#).

12.25.2.26 EI_OSABI [2/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file [elf.h](#).

12.25.2.27 EI_PAD [1/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file [elf.h](#).

12.25.2.28 EI_PAD [2/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file [elf.h](#).

12.25.2.29 EI_VERSION [1/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file [elf.h](#).

12.25.2.30 EI_VERSION [2/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file [elf.h](#).

12.25.2.31 ELFCLASSNONE [1/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file [elf.h](#).

12.25.2.32 ELFCLASSNONE [2/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file [elf.h](#).

12.25.2.33 ELFDATA2LSB [1/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file [elf.h](#).

12.25.2.34 ELFDATA2LSB [2/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file [elf.h](#).

12.25.2.35 ELFDATA2MSB [1/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file [elf.h](#).

12.25.2.36 ELFDATA2MSB [2/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file [elf.h](#).

12.25.2.37 ELFDATANONE [1/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file [elf.h](#).

12.25.2.38 ELFDATANONE [2/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file [elf.h](#).

12.25.2.39 ELFOSABI_AIX

```
#define ELFOSABI_AIX 7
```

IBM AIX.

Definition at line 181 of file [elf.h](#).

12.25.2.40 ELFOSABI_FREEBSD

```
#define ELFOSABI_FREEBSD 9
```

FreeBSD.

Definition at line 183 of file [elf.h](#).

12.25.2.41 ELFOSABI_HPUX [1/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file [elf.h](#).

12.25.2.42 ELFOSABI_HPUX [2/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file [elf.h](#).

12.25.2.43 ELFOSABI_IRIX

```
#define ELFOSABI_IRIX 8
```

SGI Irix.

Definition at line 182 of file [elf.h](#).

12.25.2.44 ELFOSABI_LINUX

```
#define ELFOSABI_LINUX 3
```

Linux.

Definition at line 179 of file [elf.h](#).

12.25.2.45 ELFOSABI_MODESTO

```
#define ELFOSABI_MODESTO 11
```

Novell Modesto.

Definition at line 185 of file [elf.h](#).

12.25.2.46 ELFOSABI_NETBSD

```
#define ELFOSABI_NETBSD 2
```

NetBSD.

Definition at line 178 of file [elf.h](#).

12.25.2.47 ELFOSABI_OPENBSD

```
#define ELFOSABI_OPENBSD 12
```

OpenBSD.

Definition at line 186 of file [elf.h](#).

12.25.2.48 ELFOSABI_SOLARIS

```
#define ELFOSABI_SOLARIS 6
```

Sun Solaris.

Definition at line 180 of file [elf.h](#).

12.25.2.49 ELFOSABI_SYSV [1/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line 322 of file [elf.h](#).

12.25.2.50 ELFOSABI_SYSV [2/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line 322 of file [elf.h](#).

12.25.2.51 ELFOSABI_TRU64

```
#define ELFOSABI_TRU64 10
```

Compaq TRU64 UNIX.

Definition at line 184 of file [elf.h](#).

12.25.2.52 EM_ARC

```
#define EM_ARC 45
```

Argonaut RISC Core, Argonaut Techn Inc.

Definition at line 230 of file [elf.h](#).

12.25.2.53 NT_VERSION

```
#define NT_VERSION 1
```

Contains a version string.

Definition at line 495 of file [elf.h](#).

12.25.2.54 PT_GNU_EH_FRAME

```
#define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
```

EH frame information.

Definition at line 458 of file [elf.h](#).

12.25.2.55 PT_GNU_RELRO

```
#define PT_GNU_RELRO (PT_LOOS + 0x474e552)
```

Read only after reloc.

Definition at line 460 of file [elf.h](#).

12.25.2.56 PT_GNU_STACK

```
#define PT_GNU_STACK (PT_LOOS + 0x474e551)
```

Flags for stack.

Definition at line 459 of file [elf.h](#).

12.25.2.57 PT_HIOS

```
#define PT_HIOS 0x6fffffff
```

os spec.

Definition at line 454 of file [elf.h](#).

12.25.2.58 PT_HIPROC

```
#define PT_HIPROC 0x7fffffff
```

processor spec.

Definition at line 456 of file [elf.h](#).

12.25.2.59 PT_L4_AUX

```
#define PT_L4_AUX (PT_LOOS + 0x14)
```

Address of the AUX structures.

Definition at line 464 of file [elf.h](#).

12.25.2.60 PT_L4_KIP

```
#define PT_L4_KIP (PT_LOOS + 0x13)
```

Address of the KIP.

Definition at line 463 of file [elf.h](#).

12.25.2.61 PT_L4_STACK

```
#define PT_L4_STACK (PT_LOOS + 0x12)
```

Address of the stack.

Definition at line 462 of file [elf.h](#).

12.25.2.62 PT_LOOS

```
#define PT_LOOS 0x60000000
```

os spec.

Definition at line 453 of file [elf.h](#).

12.25.2.63 PT_LOPROC

```
#define PT_LOPROC 0x70000000
```

processor spec.

Definition at line 455 of file [elf.h](#).

12.25.2.64 SHF_GROUP

```
#define SHF_GROUP 0x200
```

Section is member of a group.

Definition at line 408 of file [elf.h](#).

12.25.2.65 SHF_MASKOS

```
#define SHF_MASKOS 0x0ff00000
```

OS-specific.

Definition at line 410 of file [elf.h](#).

12.25.2.66 SHF_TLS

```
#define SHF_TLS 0x400
```

Section hold thread-local data.

Definition at line 409 of file [elf.h](#).

12.25.2.67 SHT_NUM

```
#define SHT_NUM 19
```

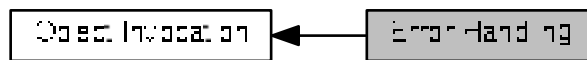
Number of defined types.

Definition at line 388 of file [elf.h](#).

12.26 Error Handling

Error handling for L4 object invocation.

Collaboration diagram for Error Handling:



Enumerations

- enum `l4_ipc_tcr_error_t` {
`L4_IPC_ERROR_MASK` = 0x1F, `L4_IPC_SND_ERR_MASK` = 0x01, `L4_IPC_ENOT_EXISTENT` = 0x04,
`L4_IPC_RETIMEOUT` = 0x03,
`L4_IPC_SETIMEOUT` = 0x02, `L4_IPC_RECANCELED` = 0x07, `L4_IPC_SECANCELED` = 0x06, `L4_IPC_REMAPFAILED` = 0x11,
`L4_IPC_SEMAPFAILED` = 0x10, `L4_IPC_RESNDPFTO` = 0x0b, `L4_IPC_SESNDPFTO` = 0x0a, `L4_IPC_RERCVPFTO` = 0x0d,
`L4_IPC_SERCVPFTO` = 0x0c, `L4_IPC_REABORTED` = 0x0f, `L4_IPC_SEABORTED` = 0x0e, `L4_IPC_REMSGCUT` = 0x09,
`L4_IPC_SEMSGCUT` = 0x08 }
Error codes in the error TCR.

Functions

- `l4_umword_t l4_ipc_error (l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW`
Get the error code for an object invocation.
- long `l4_error (l4_msgtag_t tag) L4_NOTHROW`
Return error code of a system call return message tag.
- int `l4_ipc_is_snd_error (l4_utcb_t *utcb) L4_NOTHROW`
Returns whether an error occurred in send phase of an invocation.
- int `l4_ipc_is_rcv_error (l4_utcb_t *utcb) L4_NOTHROW`
Returns whether an error occurred in receive phase of an invocation.
- int `l4_ipc_error_code (l4_utcb_t *utcb) L4_NOTHROW`
Get the error condition of the last invocation from the TCR.

12.26.1 Detailed Description

Error handling for L4 object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

12.26.2 Enumeration Type Documentation

12.26.2.1 l4_ipc_tcr_error_t

```
enum l4_ipc_tcr_error_t
```

Error codes in the *error* TCR.

The error codes are accessible via the *error* TCR, see [l4_thread_regs_t.error](#).

Enumerator

L4_IPC_ERROR_MASK	Mask for error bits.
L4_IPC_SND_ERR_MASK	Send error mask.
L4_IPC_ENOT_EXISTENT	Non-existing destination or source.
L4_IPC_RETIMEOUT	Timeout during receive operation.
L4_IPC_SETIMEOUT	Timeout during send operation.
L4_IPC_RECANCELED	Receive operation canceled.
L4_IPC_SECANCELED	Send operation canceled.
L4_IPC_REMAPFAILED	Map flexpage failed in receive operation.
L4_IPC_SEMAPFAILED	Map flexpage failed in send operation.
L4_IPC_RESNDPFTO	Send-pagefault timeout in receive operation.
L4_IPC_SESNDPFTO	Send-pagefault timeout in send operation.
L4_IPC_RERCVPFTO	Receive-pagefault timeout in receive operation.
L4_IPC_SERCVPFTO	Receive-pagefault timeout in send operation.
L4_IPC_REABORTED	Receive operation aborted.
L4_IPC_SEABORTED	Send operation aborted.
L4_IPC_REMSGCUT	Cut receive message, due to message buffer is too small.
L4_IPC_SEMSGCUT	Cut send message. due to message buffer is too small,

Definition at line 75 of file [ipc.h](#).

12.26.3 Function Documentation

12.26.3.1 l4_error()

```
long l4_error (
    l4_msgtag_t tag ) [inline]
```

Return error code of a system call return message tag.

Parameters

<i>tag</i>	System call return message type
------------	---------------------------------

Returns

0 for no error, error number in case of error

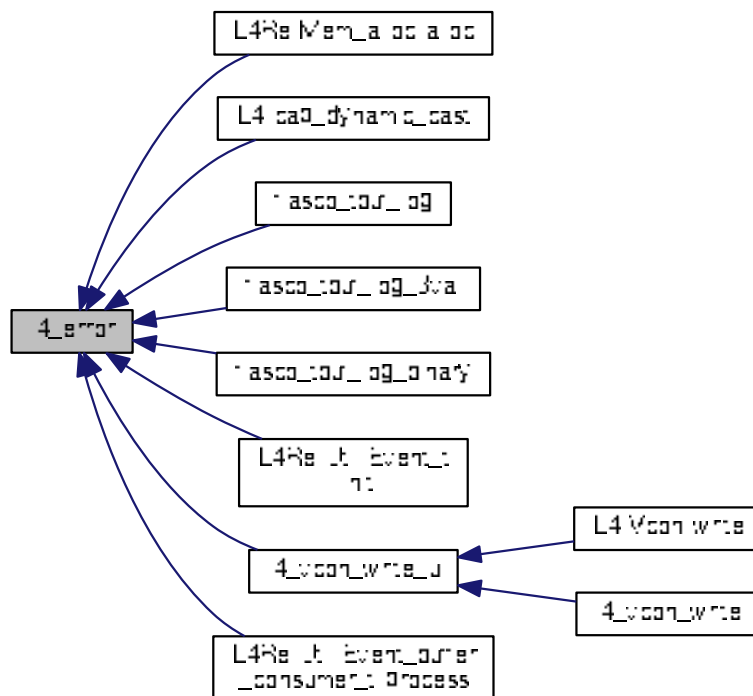
Examples:

[examples/libs/l4re/streammap/client.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/migrate/thread↵
_migrate.cc](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-
ipc/main.c](#).

Definition at line 517 of file [ipc.h](#).

Referenced by [L4Re::Mem_alloc::alloc\(\)](#), [L4::cap_dynamic_cast\(\)](#), [fiasco_tbuf_log\(\)](#), [fiasco_tbuf_log_3val\(\)](#), [fiasco_tbuf_log_binary\(\)](#), [L4Re::Util::Event_t< PAYLOAD >::init\(\)](#), [l4_vcon_write_u\(\)](#), and [L4Re::Util::Event↵
buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the caller graph for this function:



12.26.3.2 l4_ipc_error()

```
l4_umword_t l4_ipc_error (
    l4_msgtag_t tag,
    l4_utcb_t * utcb ) [inline]
```

Get the error code for an object invocation.

Parameters

<i>tag</i>	Return value of the invocation.
<i>utcb</i>	UTCB that was used for the invocation.

Returns

0 if no error condition is set, error code otherwise (see [l4_ipc_tcr_error_t](#)).

Examples:

[examples/sys/ipc/ipc_example.c](#), [examples/sys/isr/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 500 of file [ipc.h](#).

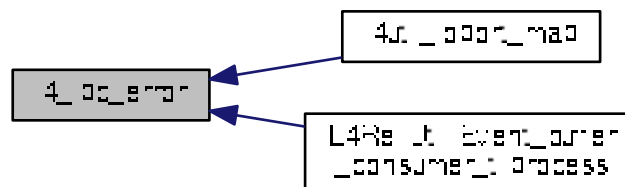
References [l4_msgtag_has_error\(\)](#).

Referenced by [l4util_ioport_map\(\)](#), and [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.26.3.3 l4_ipc_error_code()

```
int l4_ipc_error_code (
    l4\_utcb\_t * utcb ) [inline]
```

Get the error condition of the last invocation from the TCR.

Precondition

`l4_msgtag_has_error(tag) == true`

Parameters

<i>utcb</i>	UTCB to check.
-------------	----------------

Returns

Error condition of type `l4_ipc_tcr_error_t`.

Definition at line 529 of file [ipc.h](#).

12.26.3.4 l4_ipc_is_rcv_error()

```
int l4_ipc_is_rcv_error (  
    l4_utcb_t * utcb ) [inline]
```

Returns whether an error occurred in receive phase of an invocation.

Precondition

`l4_msgtag_has_error(tag) == true`

Parameters

<i>utcb</i>	UTCB to check.
-------------	----------------

Returns

Boolean value.

Definition at line 526 of file [ipc.h](#).

12.26.3.5 l4_ipc_is_snd_error()

```
int l4_ipc_is_snd_error (  
    l4_utcb_t * utcb ) [inline]
```

Returns whether an error occurred in send phase of an invocation.

Precondition

`l4_msgtag_has_error(tag) == true`

Parameters

<i>utcb</i>	UTCB to check.
-------------	----------------

Returns

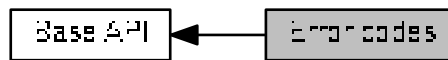
Boolean value.

Definition at line [523](#) of file [ipc.h](#).

12.27 Error codes

Common error codes.

Collaboration diagram for Error codes:



Enumerations

- enum `l4_error_code_t` {
`L4_EOK` = 0, `L4_EPERM` = 1, `L4_ENOENT` = 2, `L4_EIO` = 5,
`L4_ENXIO` = 6, `L4_E2BIG` = 7, `L4_EAGAIN` = 11, `L4_ENOMEM` = 12,
`L4_EACCESS` = 13, `L4_EFAULT` = 14, `L4_EBUSY` = 16, `L4_EEXIST` = 17,
`L4_ENODEV` = 19, `L4_EINVAL` = 22, `L4_ENOSPC` = 28, `L4_ERANGE` = 34,
`L4_ENAMETOOLONG` = 36, `L4_ENOSYS` = 38, `L4_EBADPROTO` = 39, `L4_EADDRNOTAVAIL` = 99,
`L4_ERRNOMAX` = 100, `L4_ENOREPLY` = 1000, `L4_MSGTOOSHORT` = 1001, `L4_MSGTOOLONG` = 1002,
`L4_MSGMISSARG` = 1003, `L4_EIPC_LO` = 2000, `L4_EIPC_HI` = 2000 + 0x1f }
L4 error codes.

12.27.1 Detailed Description

Common error codes.

Include File

```
#include <l4/sys/err.h>
```

12.27.2 Enumeration Type Documentation

12.27.2.1 `l4_error_code_t`

```
enum l4_error_code_t
```

L4 error codes.

Those error codes are used by both the kernel and the user programs.

Enumerator

L4_EOK	Ok.
L4_EPERM	No permission.
L4_ENOENT	No such entity.
L4_EIO	I/O error.
L4_ENXIO	No such device or address.
L4_E2BIG	Argument value too big.
L4_EAGAIN	Try again.
L4_ENOMEM	No memory.
L4_EACCESS	Permission denied.
L4_EFAULT	Invalid memory address.
L4_EBUSY	Object currently busy, try later.
L4_EEXIST	Already exists.
L4_ENODEV	No such thing.
L4_EINVAL	Invalid argument.
L4_ENOSPC	No space left on device.
L4_ERANGE	Range error.
L4_ENAMETOOLONG	Name too long.
L4_ENOSYS	No sys.
L4_EBADPROTO	Unsupported protocol.
L4_EADDRNOTAVAIL	Address not available.
L4_ERRNOMAX	Maximum error value.
L4_ENOREPLY	No reply.
L4_MSGTOOSHORT	Message too short.
L4_MSGTOOLONG	Message too long.
L4_MSGMISSARG	Message has invalid capability.
L4_EIPC_LO	Communication error-range low.
L4_EIPC_HI	Communication error-range high.

Definition at line 41 of file [err.h](#).

12.28 Event API

Event API.

Collaboration diagram for Event API:



Data Structures

- class [L4Re::Event](#)
Event class.
- class [L4Re::Event_buffer_t< PAYLOAD >](#)
Event buffer class.

12.28.1 Detailed Description

Event API.

On top of a shared [L4Re::Dataspace](#) (and optionally using [L4::Triggerable](#)), the event API implements asynchronous event transmission from an event provider (server) to an event receiver (client). Events are put into an [Event_buffer_t](#) residing on the shared [L4Re::Dataspace](#).

This interface is not usually used directly. Instead use [L4Re::Util::Event_t](#) for clients. An example server portion is implemented in [L4Re::Util::Event_svr](#).

12.29 Event interface

Event C interface.

Collaboration diagram for Event interface:



Functions

- long [l4re_event_get_buffer](#) (const [l4_cap_idx_t](#) server, const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
Get an event signal buffer.
- long [l4re_event_get_num_streams](#) (const [l4_cap_idx_t](#) server) [L4_NOTHROW](#)
Get number of streams.
- long [l4re_event_get_stream_info](#) (const [l4_cap_idx_t](#) server, int idx, [l4re_event_stream_info_t](#) *info) [L4_NOTHROW](#)
Get information on a stream.
- long [l4re_event_get_stream_info_for_id](#) (const [l4_cap_idx_t](#) server, [l4_umword_t](#) stream_id, [l4re_event_stream_info_t](#) *info) [L4_NOTHROW](#)
Get info for a stream given a stream id.
- long [l4re_event_get_axis_info](#) (const [l4_cap_idx_t](#) server, [l4_umword_t](#) id, unsigned naxes, unsigned const *axis, [l4re_event_absinfo_t](#) *info) [L4_NOTHROW](#)
Get Axis information for a stream.

12.29.1 Detailed Description

Event C interface.

12.29.2 Function Documentation

12.29.2.1 l4re_event_get_axis_info()

```

long l4re_event_get_axis_info (
    const l4\_cap\_idx\_t server,
    l4\_umword\_t id,
    unsigned naxes,
    unsigned const * axis,
    l4re\_event\_absinfo\_t * info )
  
```

Get Axis information for a stream.

Parameters

	<i>server</i>	Server to talk to.
	<i>id</i>	Id of the stream to get information from.
	<i>naxes</i>	Number of axes in <code>axis</code> array.
in	<i>axis</i>	Array of axis IDs whose information should be retrieved.
out	<i>info</i>	Information buffer to store the retrieved axis infos.

Return values

0	Success
<0	Error

See also

[L4Re::Event::get_axis_info](#)

12.29.2.2 l4re_event_get_buffer()

```
long l4re_event_get_buffer (
    const l4_cap_idx_t server,
    const l4re_ds_t ds )
```

Get an event signal buffer.

Parameters

<i>server</i>	Server to talk to.
<i>ds</i>	Buffer to event data.

Returns

0 for success, <0 on error

See also

[L4Re::Event::get_buffer](#)

12.29.2.3 l4re_event_get_num_streams()

```
long l4re_event_get_num_streams (
    const l4_cap_idx_t server )
```

Get number of streams.

Parameters

<i>server</i>	Server to talk to.
---------------	--------------------

Returns

0 for success, <0 on error

See also

L4Re::Event::get_num_streams

12.29.2.4 l4re_event_get_stream_info()

```
long l4re_event_get_stream_info (
    const l4_cap_idx_t server,
    int idx,
    l4re_event_stream_info_t * info )
```

Get information on a stream.

Parameters

<i>server</i>	Server to talk to.
<i>idx</i>	Index value.

Return values

<i>info</i>	Information buffer.
-------------	---------------------

Returns

0 for success, <0 on error

See also

L4Re::Event::get_stream_info

12.29.2.5 l4re_event_get_stream_info_for_id()

```
long l4re_event_get_stream_info_for_id (
    const l4_cap_idx_t server,
    l4_umword_t stream_id,
    l4re_event_stream_info_t * info )
```

Get info for a stream given a stream id.

Parameters

<i>server</i>	Server to talk to.
<i>stream↔ _id</i>	Stream ID.

Return values

<i>info</i>	Information buffer.
-------------	---------------------

Returns

0 for success, <0 on error

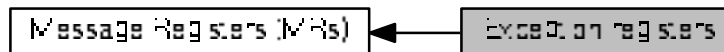
See also

L4Re::Event::get_stream_info_for_id

12.30 Exception registers

Overly definition of the MRs for exception messages.

Collaboration diagram for Exception registers:



Functions

- [l4_exc_regs_t * l4_utcb_exc](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the message-register block of a UTCB (for an exception IPC).
- [l4_umword_t l4_utcb_exc_pc](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Access function to get the program counter of the exception state.
- void [l4_utcb_exc_pc_set](#) ([l4_exc_regs_t](#) *u, [l4_addr_t](#) pc) [L4_NOTHROW](#)
Set the program counter register in the exception state.
- unsigned long [l4_utcb_exc_typeval](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Get the value out of an exception UTCB that describes the type of exception.
- int [l4_utcb_exc_is_pf](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Check whether an exception IPC is a page fault.
- [l4_addr_t l4_utcb_exc_pfa](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Function to get the [L4](#) style page fault address out of an exception.
- int [l4_utcb_exc_is_ex_regs_exception](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Check whether an exception IPC was triggered via [l4_thread_ex_regs\(\)](#).

12.30.1 Detailed Description

Overly definition of the MRs for exception messages.

12.30.2 Function Documentation

12.30.2.1 [l4_utcb_exc\(\)](#)

```
l4_exc_regs_t * l4_utcb_exc (
    void ) [inline]
```

Get the message-register block of a UTCB (for an exception IPC).

Returns

A pointer to the exception message in `u`.

Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [361](#) of file [utcb.h](#).

12.30.2.2 l4_utcb_exc_is_ex_regs_exception()

```
int l4_utcb_exc_is_ex_regs_exception (
    l4_exc_regs_t const * u ) [inline]
```

Check whether an exception IPC was triggered via [l4_thread_ex_regs\(\)](#).

Return values

0	Exception was not triggered through ex_regs.
!=0	Exception was triggered through ex_regs.

This function checks if the exception was emitted by using the L4_THREAD_EX_REGS_TRIGGER_EXCEPTION flag in an [l4_thread_ex_regs\(\)](#) call.

Definition at line 115 of file [utcb.h](#).

References [l4_utcb_exc_typeval\(\)](#).

Here is the call graph for this function:



12.30.2.3 l4_utcb_exc_is_pf()

```
int l4_utcb_exc_is_pf (
    l4_exc_regs_t const * u ) [inline]
```

Check whether an exception IPC is a page fault.

Returns

0 if not, != 0 if yes

Function to check whether an exception IPC is a page fault, also applies to I/O pagefaults.

Definition at line 105 of file [utcb.h](#).

References [l4_exc_regs_t::trapno](#).

12.30.2.4 l4_utcb_exc_pc()

```
l4_umword_t l4_utcb_exc_pc (
    l4_exc_regs_t const * u ) [inline]
```

Access function to get the program counter of the exception state.

Parameters

<i>u</i>	UTCB
----------	------

Returns

The program counter register out of the exception state.

Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 90 of file [utcb.h](#).

12.30.2.5 l4_utcb_exc_pc_set()

```
void l4_utcb_exc_pc_set (
    l4_exc_regs_t * u,
    l4_addr_t pc ) [inline]
```

Set the program counter register in the exception state.

Parameters

<i>u</i>	UTCB
<i>pc</i>	The program counter to set.

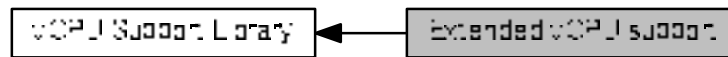
Definition at line 95 of file [utcb.h](#).

References [l4_exc_regs_t::pc](#).

12.31 Extended vCPU support

extended vCPU handling functionality.

Collaboration diagram for Extended vCPU support:



Functions

- `int l4vcpu_ext_alloc (l4_vcpu_state_t **vcpu, l4_addr_t *ext_state, l4_cap_idx_t task, l4_cap_idx_t regmgr)`
[L4_NOTHROW](#)

Allocate state area for an extended vCPU.

12.31.1 Detailed Description

extended vCPU handling functionality.

12.31.2 Function Documentation

12.31.2.1 l4vcpu_ext_alloc()

```

int l4vcpu_ext_alloc (
    l4_vcpu_state_t ** vcpu,
    l4_addr_t * ext_state,
    l4_cap_idx_t task,
    l4_cap_idx_t regmgr )

```

Allocate state area for an extended vCPU.

Return values

<code>vcpu</code>	Allocated vcpu-state area.
<code>ext_state</code>	Allocated extended vcpu-state area.

Parameters

<code>task</code>	Task to use for allocation.
-------------------	-----------------------------

Parameters

<i>regmgr</i>	Region manager to use for allocation.
---------------	---------------------------------------

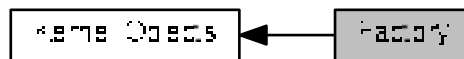
Returns

0 for success, error code otherwise

12.32 Factory

C factory interface to create kernel objects.

Collaboration diagram for Factory:



Functions

- `l4_msgtag_t l4_factory_create_task (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t const utcb_area) L4_NOTHROW`
Create a new task.
- `l4_msgtag_t l4_factory_create_task_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t const utcb_area, l4_utcb_t *utcb) L4_NOTHROW`
Create a new task.
- `l4_msgtag_t l4_factory_create_thread (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`
Create a new thread.
- `l4_msgtag_t l4_factory_create_thread_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`
Create a new thread.
- `l4_msgtag_t l4_factory_create_factory (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW`
Create a new factory.
- `l4_msgtag_t l4_factory_create_factory_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW`
Create a new factory.
- `l4_msgtag_t l4_factory_create_gate (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW`
Create a new IPC gate.
- `l4_msgtag_t l4_factory_create_gate_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label, l4_utcb_t *utcb) L4_NOTHROW`
Create a new IPC gate.
- `l4_msgtag_t l4_factory_create_irq (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`
Create a new IRQ sender.
- `l4_msgtag_t l4_factory_create_irq_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`
Create a new IRQ.
- `l4_msgtag_t l4_factory_create_vm (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`
Create a new virtual machine.
- `l4_msgtag_t l4_factory_create_vm_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`
Create a new virtual machine.

12.32.1 Detailed Description

C factory interface to create kernel objects.

A factory is used to create all kinds of kernel objects:

- [Task](#)
- [Thread](#)
- [Factory](#)
- [IPC-Gate API](#)
- [IRQs](#)
- [Virtual Machines](#)

To create a new kernel object the caller has to specify the factory to use for creation. The caller has to allocate a capability slot where the kernel stores the new object's capability.

The factory is equipped with a limit that limits the amount of kernel memory available for that factory.

Note

The limit does not give any guarantee for the amount of available kernel memory.

Include File

```
#include <l4/sys/factory.h>
```

For the C++ interface refer to [L4::Factory](#).

12.32.2 Function Documentation

12.32.2.1 l4_factory_create_factory()

```
l4_msgtag_t l4_factory_create_factory (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned long limit ) [inline]
```

Create a new factory.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new factory's capability into this slot.
	<i>limit</i>	Limit for the new factory in bytes.

Returns

Syscall return tag

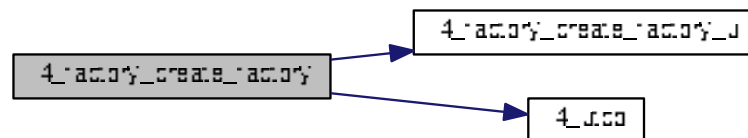
Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 373 of file [factory.h](#).

References [l4_factory_create_factory_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**12.32.2.2 l4_factory_create_factory_u()**

```

l4_msgtag_t l4_factory_create_factory_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned long limit,
    l4_utcb_t * utcb ) [inline]
  
```

Create a new factory.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new factory's capability into this slot.
	<i>limit</i>	Limit for the new factory in bytes.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

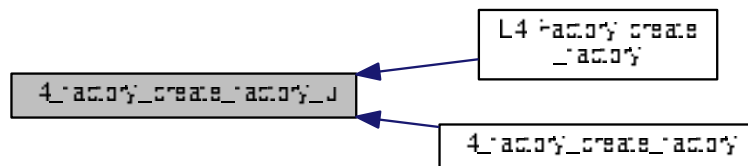
Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 307 of file [factory.h](#).

Referenced by [L4::Factory::create_factory\(\)](#), and [l4_factory_create_factory\(\)](#).

Here is the caller graph for this function:

**12.32.2.3 l4_factory_create_gate()**

```

l4_msgtag_t l4_factory_create_gate (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_cap_idx_t thread_cap,
    l4_umword_t label ) [inline]
  
```

Create a new IPC gate.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IPC gate's capability into this slot.
	<i>thread_cap</i>	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).

Returns

Syscall return tag containing one of the following return codes.

Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_ENOMEM</i>	Out-of-memory during allocation of the <i>lpc_gate</i> object.
<i>-L4_ENOENT</i>	<i>thread_cap</i> is void or points to something that is not a thread.
<i>-L4_EPERM</i>	No write rights on <i>thread_cap</i> .

An unbound IPC gate can be bound to a thread using [l4_ipc_gate_bind_thread](#).

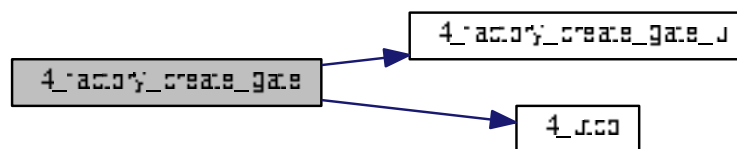
See also

[IPC-Gate API](#)

Definition at line 381 of file [factory.h](#).

References [l4_factory_create_gate_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.32.2.4 l4_factory_create_gate_u()

```

l4_msgtag_t l4_factory_create_gate_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_cap_idx_t thread_cap,
    l4_umword_t label,
    l4_utcb_t * utcb ) [inline]
  
```

Create a new IPC gate.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IPC gate's capability into this slot.
	<i>thread_cap</i>	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag containing one of the following return codes.

Return values

<code>L4_EOK</code>	No error occurred.
<code>-L4_ENOMEM</code>	Out-of-memory during allocation of the <code>lpc_gate</code> object.
<code>-L4_ENOENT</code>	<code>thread_cap</code> is void or points to something that is not a thread.
<code>-L4_EPERM</code>	No write rights on <code>thread_cap</code> .

An unbound IPC gate can be bound to a thread using `L4::lpc_gate::bind_thread()`.

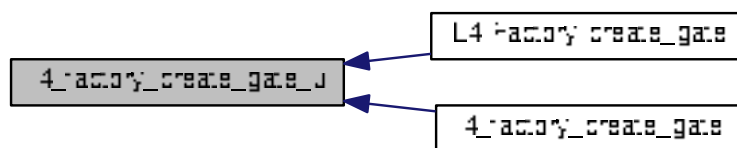
See also

[L4::lpc_gate](#)

Definition at line 318 of file [factory.h](#).

Referenced by [L4::Factory::create_gate\(\)](#), and [l4_factory_create_gate\(\)](#).

Here is the caller graph for this function:



12.32.2.5 l4_factory_create_irq()

```

l4_msgtag_t l4_factory_create_irq (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap ) [inline]
  
```

Create a new IRQ sender.

Parameters

	<i>factory</i>	Factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IRQ's capability into this slot.

Returns

Syscall return tag

See also

[IRQs](#)

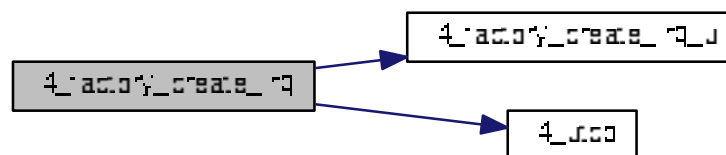
Examples:

[examples/sys/isr/main.c](#).

Definition at line 389 of file [factory.h](#).

References [l4_factory_create_irq_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.32.2.6 l4_factory_create_irq_u()

```
l4_msgtag_t l4_factory_create_irq_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [inline]
```

Create a new IRQ.

Parameters

	<i>factory</i>	Factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IRQ's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Irq>` as argument instead.

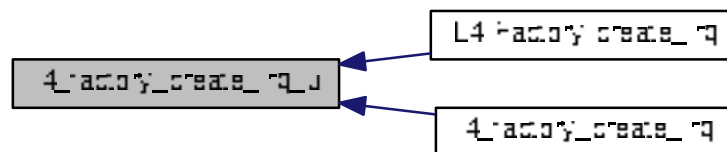
See also

[L4::Irq](#)

Definition at line 340 of file [factory.h](#).

Referenced by [L4::Factory::create_irq\(\)](#), and [l4_factory_create_irq\(\)](#).

Here is the caller graph for this function:



12.32.2.7 l4_factory_create_task()

```

l4_msgtag_t l4_factory_create_task (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_fpage_t const utcb_area ) [inline]
  
```

Create a new task.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
	<i>utcb_area</i>	Flexpage that describes an area of kernel-user memory that can be used for UTCBs and vCPU state-save-areas of the new task.

Returns

Syscall return tag.

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

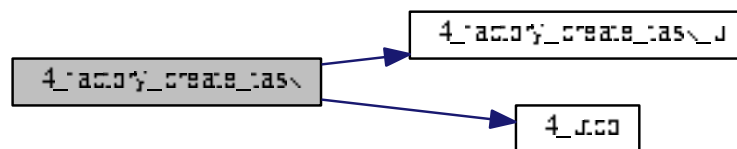
See also

[Task](#)

Definition at line 359 of file [factory.h](#).

References [l4_factory_create_task_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.32.2.8 l4_factory_create_task_u()

```

l4_msgtag_t l4_factory_create_task_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_fpage_t const utcb_area,
    l4_utcb_t * utcb ) [inline]
  
```

Create a new task.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
	<i>utcb_area</i>	Flexpage that describes an area in the address space of the new task, where the kernel should map the kernel-allocated kernel-user memory to. The kernel uses the kernel-user memory to store UTCBs and vCPU state-save-areas of the new task.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

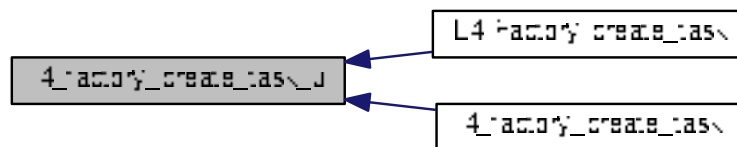
See also

[L4::Task](#)

Definition at line 289 of file [factory.h](#).

Referenced by [L4::Factory::create_task\(\)](#), and [l4_factory_create_task\(\)](#).

Here is the caller graph for this function:



12.32.2.9 l4_factory_create_thread()

```
l4_msgtag_t l4_factory_create_thread (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap ) [inline]
```

Create a new thread.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new thread's capability into this slot.

Returns

Syscall return tag

See also

[Thread](#)

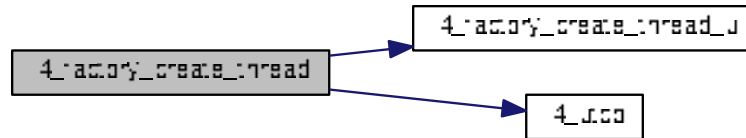
Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 366 of file [factory.h](#).

References [l4_factory_create_thread_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.32.2.10 l4_factory_create_thread_u()

```
l4_msgtag_t l4_factory_create_thread_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [inline]
```

Create a new thread.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new thread's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Thread>` as argument instead.

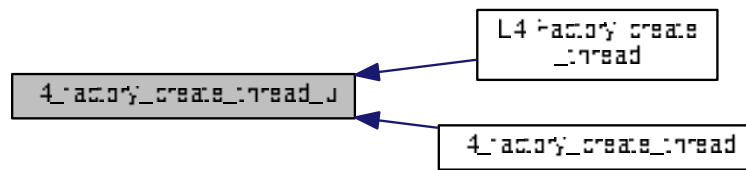
See also

[L4::Thread](#)

Definition at line 300 of file [factory.h](#).

Referenced by [L4::Factory::create_thread\(\)](#), and [l4_factory_create_thread\(\)](#).

Here is the caller graph for this function:



12.32.2.11 l4_factory_create_vm()

```
l4_msgtag_t l4_factory_create_vm (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap ) [inline]
```

Create a new virtual machine.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new VM's capability into this slot.

Returns

Syscall return tag

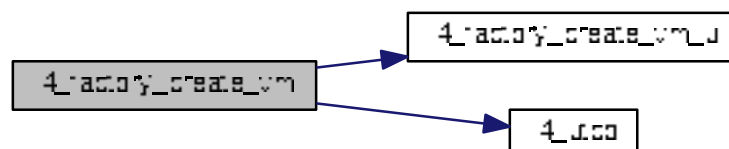
See also

[Virtual Machines](#)

Definition at line 396 of file [factory.h](#).

References [l4_factory_create_vm_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.32.2.12 l4_factory_create_vm_u()

```
l4_msgtag_t l4_factory_create_vm_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [inline]
```

Create a new virtual machine.

Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new VM's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

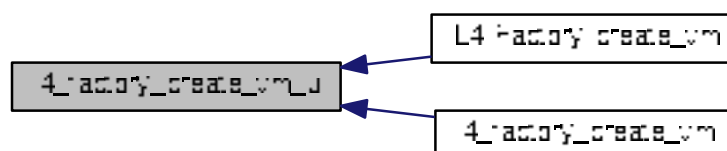
See also

[L4::Vm](#)

Definition at line 347 of file [factory.h](#).

Referenced by [L4::Factory::create_vm\(\)](#), and [l4_factory_create_vm\(\)](#).

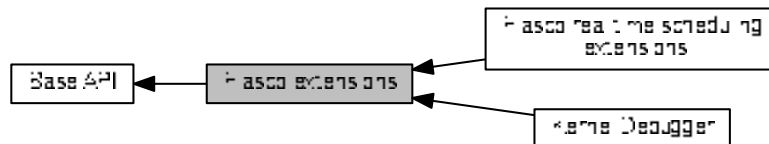
Here is the caller graph for this function:



12.33 Fiasco extensions

Kernel debugger extensions of the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco extensions:



Modules

- [Fiasco real time scheduling extensions](#)
Real time scheduling extension for the Fiasco [L4](#) implementation.
- [Kernel Debugger](#)
Kernel debugger related functionality.

Files

- file [segment.h](#)
I4f specific fs/gs manipulation
- file [segment.h](#)
I4f specific segment manipulation

Data Structures

- struct [l4_tracebuffer_status_window_t](#)
Trace-buffer status window descriptor.
- struct [l4_tracebuffer_status_t](#)
Trace-buffer status.

Enumerations

- enum {
[LOG_EVENT_CONTEXT_SWITCH](#) = 0, [LOG_EVENT_IPC_SHORTCUT](#) = 1, [LOG_EVENT_IRQ_RAISED](#) = 2, [LOG_EVENT_TIMER_IRQ](#) = 3,
[LOG_EVENT_THREAD_EX_REGS](#) = 4, [LOG_EVENT_MAX_EVENTS](#) = 16 }
Log event types.

Functions

- [l4_tracebuffer_status_t](#) * [fiasco_tbuf_get_status](#) (void)
Return trace-buffer status.
- [l4_addr_t](#) [fiasco_tbuf_get_status_phys](#) (void)
Return the physical address of the trace-buffer status struct.
- [l4_umword_t](#) [fiasco_tbuf_log](#) (const char *text)
Create new trace-buffer entry with describing <text>.
- [l4_umword_t](#) [fiasco_tbuf_log_3val](#) (const char *text, [l4_umword_t](#) v1, [l4_umword_t](#) v2, [l4_umword_t](#) v3)
Create new trace-buffer entry with describing <text> and three additional values.
- [l4_umword_t](#) [fiasco_tbuf_log_binary](#) (const unsigned char *data)
Create new trace-buffer entry with binary data.
- void [fiasco_tbuf_clear](#) (void)
Clear trace-buffer.
- void [fiasco_tbuf_dump](#) (void)
Dump trace-buffer to kernel console.
- long [fiasco_ldt_set](#) ([l4_cap_idx_t](#) task, void *ldt, unsigned int num_desc, unsigned int entry_number_start, [l4_utcb_t](#) *utcb)
Set LDT segments descriptors.
- long [fiasco_gdt_set](#) ([l4_cap_idx_t](#) thread, void *desc, unsigned int size, unsigned int entry_number_start, [l4_utcb_t](#) *utcb)
Set GDT segment descriptors.
- unsigned [fiasco_gdt_get_entry_offset](#) ([l4_cap_idx_t](#) thread, [l4_utcb_t](#) *utcb)
Return the offset of the entry in the GDT.

12.33.1 Detailed Description

Kernel debugger extensions of the Fiasco [L4](#) implementation.

12.33.2 Enumeration Type Documentation

12.33.2.1 anonymous enum

anonymous enum

Log event types.

Enumerator

LOG_EVENT_CONTEXT_SWITCH	Event: context switch.
LOG_EVENT_IPC_SHORTCUT	Event: IPC shortcut.
LOG_EVENT_IRQ_RAISED	Event: IRQ occurred.
LOG_EVENT_TIMER_IRQ	Event: Timer IRQ occurred.
LOG_EVENT_THREAD_EX_REGS	Event: thread_ex_regs.
LOG_EVENT_MAX_EVENTS	Maximum number of events.

Definition at line 37 of file [ktrace.h](#).

12.33.3 Function Documentation

12.33.3.1 fiasco_gdt_get_entry_offset()

```
unsigned fiasco_gdt_get_entry_offset (
    l4_cap_idx_t thread,
    l4_utcb_t * utcb ) [inline]
```

Return the offset of the entry in the GDT.

Parameters

<i>thread</i>	Thread to get info from.
<i>utcb</i>	UTCB of the caller.

Definition at line 149 of file [segment.h](#).

12.33.3.2 fiasco_gdt_set()

```
long fiasco_gdt_set (
    l4_cap_idx_t thread,
    void * desc,
    unsigned int size,
    unsigned int entry_number_start,
    l4_utcb_t * utcb ) [inline]
```

Set GDT segment descriptors.

Fiasco supports 3 consecutive entries, starting at the value returned by [fiasco_gdt_get_entry_offset\(\)](#)

Parameters

<i>thread</i>	Thread to set the GDT entry for.
<i>desc</i>	Pointer to GDT descriptors.
<i>size</i>	Size of the descriptors in bytes (multiple of 8).
<i>entry_number_start</i>	Entry number to start (valid values: 0-2).
<i>utcb</i>	UTCB of the caller.

Returns

System call error

Definition at line 52 of file [segment.h](#).

12.33.3.3 fiasco_ldt_set()

```
long fiasco_ldt_set (
    l4_cap_idx_t task,
    void * ldt,
    unsigned int num_desc,
    unsigned int entry_number_start,
    l4_utcb_t * utcb ) [inline]
```

Set LDT segments descriptors.

Parameters

<i>task</i>	Task to set the segment for.
<i>ldt</i>	Pointer to LDT hardware descriptors.
<i>num_desc</i>	Number of descriptors.
<i>entry_number_start</i>	Entry number to start.
<i>utcb</i>	UTCB of the caller.

Definition at line 136 of file [segment.h](#).

References [L4_EINVAL](#), and [L4_TASK_LDT_X86_MAX_ENTRIES](#).

12.33.3.4 fiasco_tbuf_get_status()

```
l4_tracebuffer_status_t* fiasco_tbuf_get_status (
    void ) [inline]
```

Return trace-buffer status.

Returns

Pointer to trace-buffer status struct.

Definition at line 35 of file [__ktrace-impl.h](#).

12.33.3.5 fiasco_tbuf_get_status_phys()

```
l4_addr_t fiasco_tbuf_get_status_phys (
    void ) [inline]
```

Return the physical address of the trace-buffer status struct.

Returns

physical address of status struct.

Definition at line 42 of file [__ktrace-impl.h](#).

12.33.3.6 fiasco_tbuf_log()

```
l4_umword_t fiasco_tbuf_log (
    const char * text ) [inline]
```

Create new trace-buffer entry with describing <text>.

Parameters

<i>text</i>	Logging text
-------------	--------------

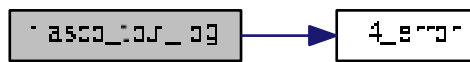
Returns

Pointer to trace-buffer entry

Definition at line 48 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

Here is the call graph for this function:



12.33.3.7 fiasco_tbuf_log_3val()

```
l4_umword_t fiasco_tbuf_log_3val (
    const char * text,
    l4_umword_t v1,
    l4_umword_t v2,
    l4_umword_t v3 ) [inline]
```

Create new trace-buffer entry with describing <text> and three additional values.

Parameters

<i>text</i>	Logging text
<i>v1</i>	first value
<i>v2</i>	second value
<i>v3</i>	third value

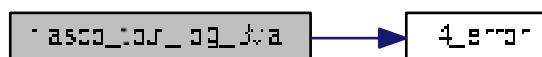
Returns

Pointer to trace-buffer entry

Definition at line 55 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

Here is the call graph for this function:

**12.33.3.8 fiasco_tbuf_log_binary()**

```
l4_umword_t fiasco_tbuf_log_binary (
    const unsigned char * data ) [inline]
```

Create new trace-buffer entry with binary data.

Parameters

<i>data</i>	binary data
-------------	-------------

Returns

Pointer to trace-buffer entry

Definition at line 78 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

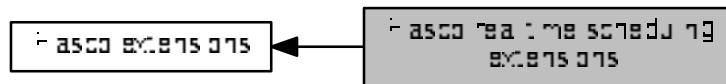
Here is the call graph for this function:



12.34 Fiasco real time scheduling extensions

Real time scheduling extension for the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco real time scheduling extensions:

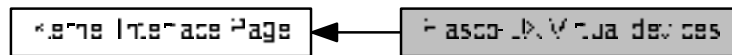


Real time scheduling extension for the Fiasco [L4](#) implementation.

12.35 Fiasco-UX Virtual devices

Virtual hardware devices, provided by Fiasco-UX.

Collaboration diagram for Fiasco-UX Virtual devices:



Data Structures

- struct [l4_vhw_entry](#)
Description of a device.
- struct [l4_vhw_descriptor](#)
Virtual hardware devices description.

Enumerations

- enum [l4_vhw_entry_type](#) { [L4_TYPE_VHW_NONE](#), [L4_TYPE_VHW_FRAMEBUFFER](#), [L4_TYPE_VHW_KEYBOARD](#), [L4_TYPE_VHW_MOUSE](#), [L4_TYPE_VHW_INPUT](#), [L4_TYPE_VHW_NET](#) }
Type of device.

12.35.1 Detailed Description

Virtual hardware devices, provided by Fiasco-UX.

Include File

```
#include <l4/sys/vhw.h>
```

12.35.2 Enumeration Type Documentation

12.35.2.1 l4_vhw_entry_type

```
enum l4_vhw_entry_type
```

Type of device.

Enumerator

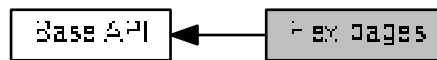
L4_TYPE_VHW_NONE	None entry.
L4_TYPE_VHW_FRAMEBUFFER	Framebuffer device.
L4_TYPE_VHW_INPUT	Input device.
L4_TYPE_VHW_NET	Network device.

Definition at line 44 of file [vhw.h](#).

12.36 Flex pages

Flex-page related API.

Collaboration diagram for Flex pages:



Data Structures

- union [l4_fpage_t](#)
L4 flexpage type.
- struct [l4_snd_fpage_t](#)
Send-flex-page types.

Enumerations

- enum [l4_fpage_consts](#) {
[L4_FPAGE_RIGHTS_SHIFT](#) = 0, [L4_FPAGE_TYPE_SHIFT](#) = 4, [L4_FPAGE_SIZE_SHIFT](#) = 6, [L4_FPAGE_ADDR_SHIFT](#) = 12,
[L4_FPAGE_RIGHTS_BITS](#) = 4, [L4_FPAGE_TYPE_BITS](#) = 2, [L4_FPAGE_SIZE_BITS](#) = 6, [L4_FPAGE_ADDR_BITS](#) = [L4_MWORD_BITS](#) - [L4_FPAGE_ADDR_SHIFT](#),
[L4_FPAGE_RIGHTS_MASK](#) }
L4 flexpage structure.
- enum { [L4_WHOLE_ADDRESS_SPACE](#) = 63 }
Constants for flexpages.
- enum [L4_fpage_rights](#) {
[L4_FPAGE_X](#) = 1, [L4_FPAGE_W](#) = 2, [L4_FPAGE_RO](#) = 4, [L4_FPAGE_RW](#) = [L4_FPAGE_RO](#) | [L4_FPAGE_W](#),
[L4_FPAGE_RX](#) = [L4_FPAGE_RO](#) | [L4_FPAGE_X](#), [L4_FPAGE_RWX](#) = [L4_FPAGE_RW](#) | [L4_FPAGE_X](#) }
Flex-page rights.
- enum [L4_cap_fpage_rights](#) {
[L4_CAP_FPAGE_W](#) = 0x1, [L4_CAP_FPAGE_S](#) = 0x2, [L4_CAP_FPAGE_R](#) = 0x4, [L4_CAP_FPAGE_RO](#) = 0x4,
[L4_CAP_FPAGE_D](#) = 0x8, [L4_CAP_FPAGE_RW](#) = [L4_CAP_FPAGE_R](#) | [L4_CAP_FPAGE_W](#), [L4_CAP_FPAGE_RS](#) = [L4_CAP_FPAGE_R](#) | [L4_CAP_FPAGE_S](#), [L4_CAP_FPAGE_RWS](#) = [L4_CAP_FPAGE_RW](#) | [L4_CAP_FPAGE_S](#),
[L4_CAP_FPAGE_RWSD](#) = [L4_CAP_FPAGE_RWS](#) | [L4_CAP_FPAGE_D](#), [L4_CAP_FPAGE_RWD](#) = [L4_CAP_FPAGE_RW](#) | [L4_CAP_FPAGE_D](#), [L4_CAP_FPAGE_RSD](#) = [L4_CAP_FPAGE_RS](#) | [L4_CAP_FPAGE_D](#) }
Cap-flex-page rights.
- enum [L4_fpage_type](#)
Flex-page type.
- enum [L4_fpage_control](#)

Flex-page map control flags.

- enum `L4_obj_fpage_ctl` {
`L4_FPAGE_C_REF_CNT` = 0x00, `L4_FPAGE_C_NO_REF_CNT` = 0x10, `L4_FPAGE_C_OBJ_RIGHT1` = 0x20, `L4_FPAGE_C_OBJ_RIGHT2` = 0x40,
`L4_FPAGE_C_OBJ_RIGHT3` = 0x80, `L4_FPAGE_C_OBJ_RIGHTS` = 0xe0, `L4_FPAGE_C_IPCGATE_SVR` = `L4_FPAGE_C_OBJ_RIGHT1` }

Flex-page map control for capabilities (snd_base)

- enum `l4_fpage_cacheability_opt_t` { `L4_FPAGE_CACHE_OPT` = 0x1, `L4_FPAGE_CACHEABLE` = 0x3, `L4_FPAGE_BUFFERABLE` = 0x5, `L4_FPAGE_UNCACHEABLE` = 0x1 }

Flex-page cacheability option.

- enum { `L4_WHOLE_IOADDRESS_SPACE` = 16, `L4_IOPORT_MAX` = (1L << `L4_WHOLE_IOADDRESS_SPACE`) }

Special constants for IO flex pages.

Functions

- `l4_fpage_t l4_fpage (l4_addr_t address, unsigned int size, unsigned char rights)` `L4_NOTHROW`
Create a memory flex page.
- `l4_fpage_t l4_fpage_all (void)` `L4_NOTHROW`
Get a flex page, describing all address spaces at once.
- `l4_fpage_t l4_fpage_invalid (void)` `L4_NOTHROW`
Get an invalid flex page.
- `l4_fpage_t l4_iofpage (unsigned long port, unsigned int size)` `L4_NOTHROW`
Create an IO-port flex page.
- `l4_fpage_t l4_obj_fpage (l4_cap_idx_t obj, unsigned int order, unsigned char rights)` `L4_NOTHROW`
Create a kernel-object flex page.
- `int l4_is_fpage_writable (l4_fpage_t fp)` `L4_NOTHROW`
Test if the flex page is writable.
- `unsigned l4_fpage_rights (l4_fpage_t f)` `L4_NOTHROW`
Return rights from a flex page.
- `unsigned l4_fpage_type (l4_fpage_t f)` `L4_NOTHROW`
Return type from a flex page.
- `unsigned l4_fpage_size (l4_fpage_t f)` `L4_NOTHROW`
Return size from a flex page.
- `unsigned long l4_fpage_page (l4_fpage_t f)` `L4_NOTHROW`
Return the page part from a flex page.
- `l4_addr_t l4_fpage_memaddr (l4_fpage_t f)` `L4_NOTHROW`
Return the memory address from the memory flex page.
- `l4_cap_idx_t l4_fpage_obj (l4_fpage_t f)` `L4_NOTHROW`
Return the capability index from the object flex page.
- `unsigned long l4_fpage_ioport (l4_fpage_t f)` `L4_NOTHROW`
Return the IO port number from the IO flex page.
- `l4_fpage_t l4_fpage_set_rights (l4_fpage_t src, unsigned char new_rights)` `L4_NOTHROW`
Set new right in a flex page.
- `int l4_fpage_contains (l4_fpage_t fpage, l4_addr_t addr, unsigned size)` `L4_NOTHROW`
Test whether a given range is completely within an fpage.
- `unsigned char l4_fpage_max_order (unsigned char order, l4_addr_t addr, l4_addr_t min_addr, l4_addr_t max_addr, l4_addr_t hotspot=0)`
Determine maximum flex page size of a region.

12.36.1 Detailed Description

Flex-page related API.

A flex page is a page with a variable size, that can describe memory, IO-Ports (IA32 only), and sets of kernel objects.

A flex page describes an always size aligned region of an address space. The size is given in a log2 scale. This means the size in elements (bytes for memory, ports for IO-Ports, and capabilities for kernel objects) is always a power of two.

A flex page also carries type and access right information for the described region. The type information selects the address space in which the flex page is valid. Access rights have a meaning depending on the specific address space (type).

There exists a special type for defining *receive windows* or for the [l4_task_unmap\(\)](#) method, that can be used to describe all address spaces (all types) with a single flex page.

12.36.2 Enumeration Type Documentation

12.36.2.1 anonymous enum

anonymous enum

Constants for flexpages.

Enumerator

L4_WHOLE_ADDRESS_SPACE	Whole address space size.
------------------------	---------------------------

Definition at line 89 of file [__l4_fpage.h](#).

12.36.2.2 anonymous enum

anonymous enum

Special constants for IO flex pages.

Enumerator

L4_WHOLE_IOADDRESS_SPACE	Whole I/O address space size.
L4_IOPORT_MAX	Maximum I/O port address.

Definition at line 281 of file [__l4_fpage.h](#).

12.36.2.3 L4_cap_fpage_rights

```
enum L4_cap_fpage_rights
```

Cap-flex-page rights.

Capabilities are modified or transfered with map and unmap operations. For that capabilities are wrapped into flex-page objects. The flex-page carries a set of rights the sender wants to hand over to the receiver along with the capability.

For the user only the 'S' and the 'W' right are visible. Other rights such as the 'D' right are internal to the corresponding kernel object and cannot be evaluated by the receiver.

Note

A thread can also map a capability from its task's capability table with a reduced set of rights into another slot of its own capability table.

Enumerator

L4_CAP_FPAGE_W	Interface specific 'W' right for capability flex-pages. The semantics of the 'W' right is defined by the protocol. For example in case of a dataspace cap, the 'W' right is needed to get a writable dataspace.
L4_CAP_FPAGE_S	Interface specific 'S' right for capability flex-pages. The semantics of the 'S' right is defined by the interface. The kernel masks this right with the 'S' right of the IPC gate over which the capability is mapped. That means that the receiver capability will only have the 'S' right set if both the flex-page and the IPC gate have the 'S' bit set.
L4_CAP_FPAGE_R	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_RO	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_D	Delete right for capability flex-pages. This allows the receiver to delete the corresponding kernel object using unmap() regardless of other tasks still holding a capability to the kernel object. Such capabilities are set to an empty capability if the object is deleted.
L4_CAP_FPAGE_RW	Read and interface specific 'W' right for capability flex-pages. The semantics of the 'W' right is defined by the interface. See also L4_CAP_FPAGE_W
L4_CAP_FPAGE_RS	Read and interface specific 'S' right for capability flex-pages. The semantics of the 'S' right is defined by the interface. See also L4_CAP_FPAGE_S
L4_CAP_FPAGE_RWS	Read, interface specific 'W', and 'S' rights for capability flex-pages. The semantics of the 'W' and 'S' right are defined by the interface. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , and L4_CAP_FPAGE_S

Enumerator

L4_CAP_FPAGE_RWSD	Full rights for capability flex-pages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , L4_CAP_FPAGE_S , and L4_CAP_FPAGE_D
L4_CAP_FPAGE_RWD	Read, write, and delete right for capability flex-pages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , and L4_CAP_FPAGE_D
L4_CAP_FPAGE_RSD	Read, 'S', and delete right for capability flex-pages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_S , and L4_CAP_FPAGE_D

Definition at line 134 of file [__l4_fpage.h](#).

12.36.2.4 l4_fpage_cacheability_opt_t

```
enum l4_fpage_cacheability_opt_t
```

Flex-page cacheability option.

Enumerator

L4_FPAGE_CACHE_OPT	Enable the cacheability option in a send flex page.
L4_FPAGE_CACHEABLE	Cacheability option to enable caches for the mapping.
L4_FPAGE_BUFFERABLE	Cacheability option to enable buffered writes for the mapping.
L4_FPAGE_UNCACHEABLE	Cacheability option to disable caching for the mapping.

Definition at line 262 of file [__l4_fpage.h](#).

12.36.2.5 l4_fpage_consts

```
enum l4_fpage_consts
```

[L4](#) flexpage structure.

Enumerator

L4_FPAGE_RIGHTS_SHIFT	Access permissions shift.
L4_FPAGE_TYPE_SHIFT	Flexpage type shift (memory, IO port, obj...)
L4_FPAGE_SIZE_SHIFT	Flexpage size shift (log2-based)

Enumerator

L4_FPAGE_ADDR_SHIFT	Page address shift.
L4_FPAGE_RIGHTS_BITS	Access permissions size.
L4_FPAGE_TYPE_BITS	Flexpage type size (memory, IO port, obj...)
L4_FPAGE_SIZE_BITS	Flexpage size size (log2-based)
L4_FPAGE_ADDR_BITS	Page address size.
L4_FPAGE_RIGHTS_MASK	Mask to get the flexpage rights.

Definition at line 55 of file [__l4_fpage.h](#).

12.36.2.6 L4_fpage_rights

```
enum L4_fpage_rights
```

Flex-page rights.

Enumerator

L4_FPAGE_X	Executable flex page.
L4_FPAGE_W	Writable flex page.
L4_FPAGE_RO	Read-only flex page.
L4_FPAGE_RW	Read-write flex page.
L4_FPAGE_RX	Read-execute flex page.
L4_FPAGE_RWX	Read-write-execute flex page.

Definition at line 107 of file [__l4_fpage.h](#).

12.36.2.7 L4_obj_fpage_ctl

```
enum L4_obj_fpage_ctl
```

Flex-page map control for capabilities (snd_base)

These rights need to be added to the snd_base when mapping and control internal behavior. The exact meaning depends on the type of capability (currently used only with IPC gates).

Enumerator

L4_FPAGE_C_REF_CNT	Mapping is reference-counted (default).
L4_FPAGE_C_NO_REF_CNT	Don't increase the reference counter.
L4_FPAGE_C_OBJ_RIGHT1	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT2	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT3	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHTS	All Object-type specific right bits.
L4_FPAGE_C_IPCGATE_SVR Generated for L4Re by Doxygen	The receiver may invoke IPC-gate-specific functions on the capability, e.g. bind a thread to the gate and modify the label. Needed if the receiver implements the server side of an IPC gate.

Definition at line 240 of file [__l4_fpage.h](#).

12.36.3 Function Documentation

12.36.3.1 l4_fpage()

```
l4_fpage_t l4_fpage (
    l4_addr_t address,
    unsigned int size,
    unsigned char rights ) [inline]
```

Create a memory flex page.

Parameters

<i>address</i>	Flex-page start address
<i>size</i>	Flex-page size (log2), L4_WHOLE_ADDRESS_SPACE to specify the whole address space (with address 0)
<i>rights</i>	Access rights, see L4_fpage_rights

Returns

Memory flex page

Definition at line 633 of file [__l4_fpage.h](#).

Referenced by [L4Re::Rm::attach\(\)](#).

Here is the caller graph for this function:



12.36.3.2 l4_fpage_all()

```
l4_fpage_t l4_fpage_all (
    void ) [inline]
```

Get a flex page, describing all address spaces at once.

Returns

Special *all-spaces* flex page.

Definition at line 653 of file [__l4_fpage.h](#).

References [L4_WHOLE_ADDRESS_SPACE](#).

12.36.3.3 l4_fpage_contains()

```
int l4_fpage_contains (
    l4_fpage_t fpage,
    l4_addr_t addr,
    unsigned size ) [inline]
```

Test whether a given range is completely within an fpage.

Parameters

<i>fpage</i>	Flex page
<i>addr</i>	Address
<i>size</i>	Size of range in log2.

Return values

<i>==0</i>	The range is not completely in the fpage.
<i>!=0</i>	The range is within the fpage.

Definition at line 685 of file [__l4_fpage.h](#).

References [l4_fpage_memaddr\(\)](#).

Here is the call graph for this function:



12.36.3.4 l4_fpage_invalid()

```
l4_fpage_t l4_fpage_invalid (
    void ) [inline]
```

Get an invalid flex page.

Returns

Special *invalid* flex page.

Definition at line 659 of file [__l4_fpage.h](#).

12.36.3.5 l4_fpage_ioport()

```
unsigned long l4_fpage_ioport (
    l4_fpage_t f ) [inline]
```

Return the IO port number from the IO flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

IO port number from the given IO flex page.

Precondition

f must be an IO flex page (`l4_fpage_type(f) == L4_FPAGE_IO`) and

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 589 of file [__l4_fpage.h](#).

12.36.3.6 l4_fpage_max_order()

```
unsigned char l4_fpage_max_order (
    unsigned char order,
    l4_addr_t addr,
    l4_addr_t min_addr,
    l4_addr_t max_addr,
    l4_addr_t hotspot = 0 ) [inline]
```

Determine maximum flex page size of a region.

Parameters

<i>order</i>	Order value to start with (e.g. for memory L4_LOG2_PAGESIZE would be used)
<i>addr</i>	Address to be covered by the flex page.
<i>min_addr</i>	Start of region / minimal address (including).
<i>max_addr</i>	End of region / maximal address (excluding).
<i>hotspot</i>	(Optional) hot spot.

Returns

Maximum order (log2-size) possible.

Note

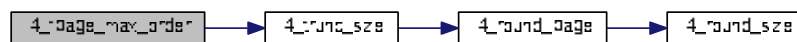
The start address of the flex-page can be determined with `l4_trunc_size(addr, returnvalue)`

Definition at line 693 of file `__l4_fpage.h`.

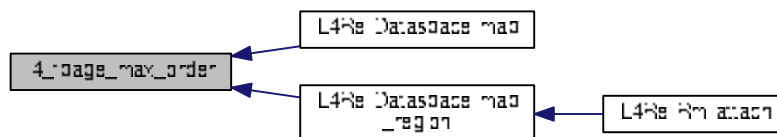
References `l4_trunc_size()`.

Referenced by `L4Re::Dataspace::map()`, and `L4Re::Dataspace::map_region()`.

Here is the call graph for this function:



Here is the caller graph for this function:



12.36.3.7 l4_fpage_memaddr()

```

l4_addr_t l4_fpage_memaddr (
    l4_fpage_t f ) [inline]
  
```

Return the memory address from the memory flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Page address from the given memory flex page.

Precondition

f must be a memory flex page (`l4_fpage_type(f) == L4_FPAGE_MEMORY`).

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 595 of file `__l4_fpage.h`.

Referenced by `l4_fpage_contains()`.

Here is the caller graph for this function:

**12.36.3.8 l4_fpage_obj()**

```
l4_cap_idx_t l4_fpage_obj (
    l4_fpage_t f ) [inline]
```

Return the capability index from the object flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Capability index from the given object flex page.

Precondition

`f` must be an object flex page (`l4_fpage_type(f) == L4_FPAGE_OBJ`)

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 601 of file [__l4_fpage.h](#).

12.36.3.9 l4_fpage_page()

```
unsigned long l4_fpage_page (
    l4_fpage_t f ) [inline]
```

Return the page part from a flex page.

Parameters

<code>f</code>	Flex page
----------------	-----------

Returns

Page part of the given flex page.

Note

The meaning of the page part depends on the flex-page type.

Definition at line 583 of file [__l4_fpage.h](#).

12.36.3.10 l4_fpage_rights()

```
unsigned l4_fpage_rights (
    l4_fpage_t f ) [inline]
```

Return rights from a flex page.

Parameters

<code>f</code>	Flex page
----------------	-----------

Returns

Size part of the given flex page.

Definition at line 565 of file [__l4_fpage.h](#).

References [L4_FPAGE_RIGHTS_MASK](#), and [L4_FPAGE_RIGHTS_SHIFT](#).

Referenced by [l4_is_fpage_writable\(\)](#).

Here is the caller graph for this function:



12.36.3.11 l4_fpage_set_rights()

```
l4_fpage_t l4_fpage_set_rights (
    l4_fpage_t src,
    unsigned char new_rights ) [inline]
```

Set new right in a flex page.

Parameters

<i>src</i>	Flex page
<i>new_rights</i>	New rights

Returns

Modified flex page with new rights.

Definition at line 624 of file [__l4_fpage.h](#).

References [l4_fpage_t::raw](#).

12.36.3.12 l4_fpage_size()

```
unsigned l4_fpage_size (
    l4_fpage_t f ) [inline]
```

Return size from a flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Size part of the given flex page.

See also

[l4_fpage_memaddr\(\)](#), [l4_fpage_obj\(\)](#), [l4_fpage_ioport\(\)](#)

Definition at line 577 of file [__l4_fpage.h](#).

12.36.3.13 l4_fpage_type()

```
unsigned l4_fpage_type (
    l4_fpage_t f ) [inline]
```

Return type from a flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Type part of the given flex page.

Definition at line 571 of file [__l4_fpage.h](#).

12.36.3.14 l4_iofpage()

```
l4_fpage_t l4_iofpage (
    unsigned long port,
    unsigned int size ) [inline]
```

Create an IO-port flex page.

Parameters

<i>port</i>	I/O-flex-page port base
<i>size</i>	I/O-flex-page size (log2), L4_WHOLE_IOADDRESS_SPACE to specify the whole I/O address space (with <code>port 0</code>)

Returns

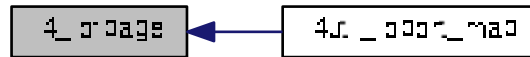
I/O flex page

Definition at line 639 of file [__l4_fpage.h](#).

References [L4_FPAGE_ADDR_SHIFT](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



12.36.3.15 l4_is_fpage_writable()

```
int l4_is_fpage_writable (
    l4_fpage_t fp ) [inline]
```

Test if the flex page is writable.

Parameters

<i>fp</i>	Flex page.
-----------	------------

Return values

<i>!=0</i>	if flex page is writable.
<i>==0</i>	if flex page is not writable.

Definition at line 666 of file [__l4_fpage.h](#).

References [l4_fpage_rights\(\)](#), and [L4_FPAGE_W](#).

Here is the call graph for this function:



12.36.3.16 l4_obj_fpage()

```
l4_fpage_t l4_obj_fpage (
    l4_cap_idx_t obj,
    unsigned int order,
    unsigned char rights ) [inline]
```

Create a kernel-object flex page.

Parameters

<i>obj</i>	Base capability selector.
<i>order</i>	Log2 size (number of capabilities).
<i>rights</i>	Access rights, see L4_cap_fpage_rights

Returns

Flex page for a set of kernel objects.

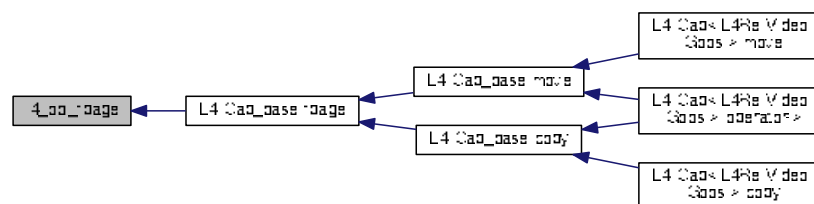
Note

[L4_CAP_FPAGE_R](#) is always required, otherwise no capability is mapped.

Definition at line 645 of file [__l4_fpage.h](#).

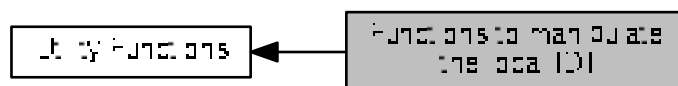
Referenced by [L4::Cap_base::fpage\(\)](#).

Here is the caller graph for this function:



12.37 Functions to manipulate the local IDT

Collaboration diagram for Functions to manipulate the local IDT:



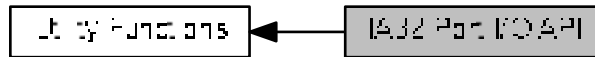
Data Structures

- struct [l4util_idt_desc_t](#)
IDT entry.
- struct [l4util_idt_header_t](#)
Header of an IDT table.

12.37.1 Detailed Description

12.38 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:



Functions

- [l4_uint8_t l4util_in8 \(l4_uint16_t port\)](#)
Read byte from I/O port.
- [l4_uint16_t l4util_in16 \(l4_uint16_t port\)](#)
Read 16-bit-value from I/O port.
- [l4_uint32_t l4util_in32 \(l4_uint16_t port\)](#)
Read 32-bit-value from I/O port.
- [void l4util_ins8 \(l4_uint16_t port, l4_umword_t addr, l4_umword_t count\)](#)
Read a block of 8-bit-values from I/O ports.
- [void l4util_ins16 \(l4_uint16_t port, l4_umword_t addr, l4_umword_t count\)](#)
Read a block of 16-bit-values from I/O ports.
- [void l4util_ins32 \(l4_uint16_t port, l4_umword_t addr, l4_umword_t count\)](#)
Read a block of 32-bit-values from I/O ports.
- [void l4util_out8 \(l4_uint8_t value, l4_uint16_t port\)](#)
Write byte to I/O port.
- [void l4util_out16 \(l4_uint16_t value, l4_uint16_t port\)](#)
Write 16-bit-value to I/O port.
- [void l4util_out32 \(l4_uint32_t value, l4_uint16_t port\)](#)
Write 32-bit-value to I/O port.
- [void l4util_outs8 \(l4_uint16_t port, l4_umword_t addr, l4_umword_t count\)](#)
Write a block of bytes to I/O port.
- [void l4util_outs16 \(l4_uint16_t port, l4_umword_t addr, l4_umword_t count\)](#)
Write a block of 16-bit-values to I/O port.
- [void l4util_outs32 \(l4_uint16_t port, l4_umword_t addr, l4_umword_t count\)](#)
Write block of 32-bit-values to I/O port.
- [void l4util_iodelay \(void\)](#)
delay I/O port access by writing to port 0x80

12.38.1 Detailed Description

12.38.2 Function Documentation

12.38.2.1 l4util_in16()

```
l4_uint16_t l4util_in16 (  
    l4_uint16_t port ) [inline]
```

Read 16-bit-value from I/O port.

Parameters

<i>port</i>	I/O port address
-------------	------------------

Returns

value

Definition at line 180 of file [port_io.h](#).

12.38.2.2 l4util_in32()

```
l4_uint32_t l4util_in32 (  
    l4_uint16_t port ) [inline]
```

Read 32-bit-value from I/O port.

Parameters

<i>port</i>	I/O port address
-------------	------------------

Returns

value

Definition at line 188 of file [port_io.h](#).

12.38.2.3 l4util_in8()

```
l4_uint8_t l4util_in8 (  
    l4_uint16_t port ) [inline]
```

Read byte from I/O port.

Parameters

<i>port</i>	I/O port address
-------------	------------------

Returns

value

Definition at line 172 of file [port_io.h](#).

Referenced by [l4util_irq_acknowledge\(\)](#).

Here is the caller graph for this function:



12.38.2.4 l4util_ins16()

```
void l4util_ins16 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Read a block of 16-bit-values from I/O ports.

Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 205 of file [port_io.h](#).

12.38.2.5 l4util_ins32()

```
void l4util_ins32 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Read a block of 32-bit-values from I/O ports.

Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 214 of file [port_io.h](#).

12.38.2.6 l4util_ins8()

```
void l4util_ins8 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Read a block of 8-bit-values from I/O ports.

Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 196 of file [port_io.h](#).

12.38.2.7 l4util_out16()

```
void l4util_out16 (
    l4_uint16_t value,
    l4_uint16_t port ) [inline]
```

Write 16-bit-value to I/O port.

Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 229 of file [port_io.h](#).

12.38.2.8 l4util_out32()

```
void l4util_out32 (
    l4_uint32_t value,
    l4_uint16_t port ) [inline]
```

Write 32-bit-value to I/O port.

Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 235 of file [port_io.h](#).

12.38.2.9 l4util_out8()

```
void l4util_out8 (
    l4_uint8_t value,
    l4_uint16_t port ) [inline]
```

Write byte to I/O port.

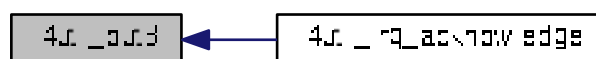
Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 223 of file [port_io.h](#).

Referenced by [l4util_irq_acknowledge\(\)](#).

Here is the caller graph for this function:



12.38.2.10 l4util_outs16()

```
void l4util_outs16 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Write a block of 16-bit-values to I/O port.

Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 250 of file [port_io.h](#).

12.38.2.11 l4util_outs32()

```
void l4util_outs32 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Write block of 32-bit-values to I/O port.

Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 259 of file [port_io.h](#).

12.38.2.12 l4util_outs8()

```
void l4util_outs8 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count ) [inline]
```

Write a block of bytes to I/O port.

Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 241 of file [port_io.h](#).

12.39 IO interface

Typedefs

- typedef `l4vbus_resource_t` `l4io_resource_t`
Resource descriptor.
- typedef `l4vbus_device_t` `l4io_device_t`
Device descriptor.

Enumerations

- enum `l4io_iomem_flags_t` {
`L4IO_MEM_NONCACHED` = 0, `L4IO_MEM_CACHED` = 1, `L4IO_MEM_USE_MTRR` = 2, `L4IO_MEM_USE_RESERVED_AREA` = 0x40 << 8,
`L4IO_MEM_EAGER_MAP` = 0x80 << 8 }
Flags for IO memory.
- enum `l4io_device_types_t` {
`L4IO_DEVICE_INVALID` = 0, `L4IO_DEVICE_PCI`, `L4IO_DEVICE_USB`, `L4IO_DEVICE_OTHER`,
`L4IO_DEVICE_ANY` = ~0 }
Device types.
- enum `l4io_resource_types_t` {
`L4IO_RESOURCE_INVALID` = `L4VBUS_RESOURCE_INVALID`, `L4IO_RESOURCE_IRQ` = `L4VBUS_RESOURCE_IRQ`, `L4IO_RESOURCE_MEM` = `L4VBUS_RESOURCE_MEM`, `L4IO_RESOURCE_PORT` = `L4VBUS_RESOURCE_PORT`,
`L4IO_RESOURCE_ANY` = ~0 }
Resource types.

Functions

- long `l4io_request_iomem` (`l4_addr_t` phys, unsigned long size, int flags, `l4_addr_t` *virt)
Request an IO memory region.
- long `l4io_request_iomem_region` (`l4_addr_t` phys, `l4_addr_t` virt, unsigned long size, int flags)
Request an IO memory region and map it to a specified region.
- long `l4io_release_iomem` (`l4_addr_t` virt, unsigned long size)
Release an IO memory region.
- long `l4io_search_iomem_region` (`l4_addr_t` phys, `l4_addr_t` size, `l4_addr_t` *rstart, `l4_addr_t` *rsize)
Search for a IO memory region.
- long `l4io_request_ioport` (unsigned portnum, unsigned len)
Request an IO port region.
- long `l4io_release_ioport` (unsigned portnum, unsigned len)
Release an IO port region.
- int `l4io_lookup_device` (const char *devname, `l4io_device_handle_t` *dev_handle, `l4io_device_t` *dev, `l4io_resource_handle_t` *res_handle)
Find a device by name.
- int `l4io_lookup_resource` (`l4io_device_handle_t` devhandle, enum `l4io_resource_types_t` type, `l4io_resource_handle_t` *reshandle, `l4io_resource_t` *res)
Request a specific resource from a device description.
- `l4_addr_t` `l4io_request_resource_iomem` (`l4io_device_handle_t` devhandle, `l4io_resource_handle_t` *reshandle)
Request IO memory.
- int `l4io_has_resource` (enum `l4io_resource_types_t` type, `l4vbus_paddr_t` start, `l4vbus_paddr_t` end)
Check if a resource is available.

12.39.1 Detailed Description

12.39.2 Typedef Documentation

12.39.2.1 l4io_resource_t

```
typedef l4vbus_resource_t l4io_resource_t
```

Resource descriptor.

For IRQ types, the end field is not used, i.e. only a single interrupt can be described with a l4io_resource_t

Definition at line 69 of file [types.h](#).

12.39.3 Enumeration Type Documentation

12.39.3.1 l4io_device_types_t

```
enum l4io_device_types_t
```

Device types.

Enumerator

L4IO_DEVICE_INVALID	Invalid type.
L4IO_DEVICE_PCI	PCI device.
L4IO_DEVICE_USB	USB device.
L4IO_DEVICE_OTHER	Any other device without unique IDs.
L4IO_DEVICE_ANY	any type

Definition at line 38 of file [types.h](#).

12.39.3.2 l4io_iomem_flags_t

```
enum l4io_iomem_flags_t
```

Flags for IO memory.

Enumerator

L4IO_MEM_NONCACHED	Non-cache memory.
--------------------	-------------------

Enumerator

L4IO_MEM_CACHED	Cache memory.
L4IO_MEM_USE_MTRR	Use MTRR.
L4IO_MEM_USE_RESERVED_AREA	Use reserved area for mapping I/O memory. Flag only valid for l4io_request_iomem_region()
L4IO_MEM_EAGER_MAP	Eagerly map the I/O memory. Passthrough to the l4re-rm.

Definition at line 16 of file [types.h](#).

12.39.3.3 l4io_resource_types_t

```
enum l4io_resource_types_t
```

Resource types.

Enumerator

L4IO_RESOURCE_INVALID	Invalid type.
L4IO_RESOURCE_IRQ	Interrupt resource.
L4IO_RESOURCE_MEM	I/O memory resource.
L4IO_RESOURCE_PORT	I/O port resource (x86 only)
L4IO_RESOURCE_ANY	any type

Definition at line 50 of file [types.h](#).

12.39.4 Function Documentation

12.39.4.1 l4io_has_resource()

```
int l4io_has_resource (
    enum l4io_resource_types_t type,
    l4vbus_paddr_t start,
    l4vbus_paddr_t end )
```

Check if a resource is available.

Parameters

<i>type</i>	Type of resource
<i>start</i>	Minimal value.
<i>end</i>	Maximum value.

12.39.4.2 l4io_lookup_device()

```
int l4io_lookup_device (
    const char * devname,
    l4io_device_handle_t * dev_handle,
    l4io_device_t * dev,
    l4io_resource_handle_t * res_handle )
```

Find a device by name.

Parameters

<i>devname</i>	Name of device
----------------	----------------

Return values

<i>dev_handle</i>	Device handle for found device, can be NULL.
<i>dev</i>	Device information, filled by the function, can be NULL.
<i>res_handle</i>	Resource handle, can be NULL.

Returns

0 on success, error code otherwise

12.39.4.3 l4io_lookup_resource()

```
int l4io_lookup_resource (
    l4io_device_handle_t devhandle,
    enum l4io_resource_types_t type,
    l4io_resource_handle_t * reshandle,
    l4io_resource_t * res )
```

Request a specific resource from a device description.

Parameters

<i>devhandle</i>	Device handle.
<i>type</i>	Type of resource to request (see #l4io_resource_types_t)
<i>reshandle</i>	Resource handle, start with handle returned by device functions.

Return values

<i>reshandle</i>	Next resource handle.
<i>res</i>	Device descriptor

Returns

0 on success, error code otherwise, esp. -L4_ENOENT if no more resources found

12.39.4.4 l4io_release_iomem()

```
long l4io_release_iomem (
    l4_addr_t virt,
    unsigned long size )
```

Release an IO memory region.

Parameters

<i>virt</i>	Virtual address of region to free, see l4io_request_iomem
<i>size</i>	Size of the region to release.

Returns

0 on success, <0 on error

12.39.4.5 l4io_release_ioport()

```
long l4io_release_ioport (
    unsigned portnum,
    unsigned len )
```

Release an IO port region.

Parameters

<i>portnum</i>	Start of port range to release
<i>len</i>	Length of range to request

Returns

0 on success, <0 on error

Note

X86 architecture only

12.39.4.6 `l4io_request_iomem()`

```
long l4io_request_iomem (
    l4_addr_t phys,
    unsigned long size,
    int flags,
    l4_addr_t * virt )
```

Request an IO memory region.

Parameters

	<i>phys</i>	Physical address of the I/O memory region
	<i>size</i>	Size of the region in Bytes, granularity pages.
	<i>flags</i>	See l4io_iomem_flags_t
<i>in, out</i>	<i>virt</i>	Virtual address where the IO memory region should be mapped to. If the caller passes '0' a region in the caller's address space is searched and the virtual address is returned.

Return values

<i>0</i>	Success.
<i>-L4_ENOENT</i>	No area in the caller's address space could be found to map the IO memory region.
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	Invalid value.
<i>-L4_EADDRNOTAVAIL</i>	The requested virtual address is not available.
<i>-L4_ENOMEM</i>	The requested IO memory region could not be allocated.
<i><0</i>	IPC errors.

Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

12.39.4.7 `l4io_request_iomem_region()`

```
long l4io_request_iomem_region (
    l4_addr_t phys,
    l4_addr_t virt,
    unsigned long size,
    int flags )
```

Request an IO memory region and map it to a specified region.

Parameters

<i>phys</i>	Physical address of the I/O memory region
<i>virt</i>	Virtual address.
<i>size</i>	Size of the region in Bytes, granularity pages.
<i>flags</i>	See l4io_iomem_flags_t

Return values

<i>0</i>	Success.
<i>-L4_ENOENT</i>	No area could be found to map the IO memory region.
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	Invalid value.
<i>-L4_EADDRNOTAVAIL</i>	The requested virtual address is not available.
<i>-L4_ENOMEM</i>	The requested IO memory region could not be allocated.
<i><0</i>	IPC errors.

Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

12.39.4.8 `l4io_request_ioport()`

```
long l4io_request_ioport (
    unsigned portnum,
    unsigned len )
```

Request an IO port region.

Parameters

<i>portnum</i>	Start of port range to request
<i>len</i>	Length of range to request

Returns

0 on success, <0 on error

Note

X86 architecture only

12.39.4.9 `l4io_request_resource_iomem()`

```
l4\_addr\_t l4io_request_resource_iomem (
    l4io_device_handle_t devhandle,
    l4io_resource_handle_t * reshandle )
```

Request IO memory.

Parameters

	<i>devhandle</i>	Device handle.
<i>in, out</i>	<i>reshandle</i>	Resource handle from which IO memory should be requested. Upon successful completion 'reshandle' points to the device's next resource.

Return values

<i>0</i>	An error occurred. The value of 'reshandle' is undefined.
<i>>0</i>	The virtual address of the IO memory mapping.

12.39.4.10 `l4io_search_iomem_region()`

```
long l4io_search_iomem_region (
    l4_addr_t phys,
    l4_addr_t size,
    l4_addr_t * rstart,
    l4_addr_t * rsize )
```

Search for a IO memory region.

Parameters

<i>phys</i>	Physical address to look for
<i>size</i>	Size of requested memory area

Return values

<i>rstart</i>	Start address for region
<i>rsize</i>	Size of region in bytes

Returns

0 if an IO region was found, <0 if not

12.40 IPC-Gate API

Secure communication object.

Collaboration diagram for IPC-Gate API:



Functions

- [l4_msgtag_t l4_ipc_gate_bind_thread](#) ([l4_cap_idx_t](#) gate, [l4_cap_idx_t](#) thread, [l4_umword_t](#) label)
Bind the IPC gate to a thread.
- [l4_msgtag_t l4_ipc_gate_get_infos](#) ([l4_cap_idx_t](#) gate, [l4_umword_t](#) *label)
Get information about the IPC-gate.
- [l4_msgtag_t l4_rcv_ep_bind_thread](#) ([l4_cap_idx_t](#) ep, [l4_cap_idx_t](#) thread, [l4_umword_t](#) label)
Bind the IPC gate to a thread.

12.40.1 Detailed Description

Secure communication object.

IPC-Gate objects provide a means to establish secure communication channels to [L4 Threads](#) ([Thread](#)). An IPC-Gate object can be created using a [Factory](#) ([l4_factory_create_gate\(\)](#)) and get assigned a specific [L4](#) thread and a *label* as protected payload. The *label* has the size of one machine word and can only be seen by the Task running the thread that is assigned of the IPC-gate. The *label* is received as part of the IPC message. The *label* can thus be used to securely identify the IPC-gate that was used to send a message.

An IPC-gate is usually used to represent an user-level object and may be the address of the data structure for the object in the server task.

With client privileges an IPC-gate does not provide any direct API and thus an IPC-gate kernel object cannot be modified by invocations. Each invocation of an IPC-gate kernel object is translated into an IPC message to the assigned thread.

Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::ipc_gate](#) documentation.

See also

[Object Invocation](#)

12.40.2 Function Documentation

12.40.2.1 l4_ipc_gate_bind_thread()

```
l4_msgtag_t l4_ipc_gate_bind_thread (
    l4_cap_idx_t gate,
    l4_cap_idx_t thread,
    l4_umword_t label ) [inline]
```

Bind the IPC gate to a thread.

Parameters

<i>gate</i>	The IPC gate object.
<i>thread</i>	The thread object that shall be bound to <i>gate</i> .
<i>label</i>	Label to assign to <i>gate</i> . The two least significant bits should usually be set to zero.

Returns

Syscall return tag containing one of the following return codes.

Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<i>thread</i> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	<i>thread</i> is missing L4_CAP_FPAGE_S right.

Definition at line 130 of file [ipc_gate.h](#).

12.40.2.2 l4_ipc_gate_get_infos()

```
l4_msgtag_t l4_ipc_gate_get_infos (
    l4_cap_idx_t gate,
    l4_umword_t * label ) [inline]
```

Get information about the IPC-gate.

Parameters

	<i>gate</i>	The IPC gate object to get information about.
out	<i>label</i>	The label of the IPC gate is returned here.

Returns

System call return tag.

Definition at line 137 of file [ipc_gate.h](#).

12.40.2.3 l4_rcv_ep_bind_thread()

```
l4_msgtag_t l4_rcv_ep_bind_thread (
    l4_cap_idx_t ep,
    l4_cap_idx_t thread,
    l4_umword_t label ) [inline]
```

Bind the IPC gate to a thread.

Parameters

<i>ep</i>	The IPC receive endpoint object.
<i>thread</i>	The thread object that shall be bound to <i>ep</i> .
<i>label</i>	Label to assign to <i>ep</i> . The two least significant bits should usually be set to zero.

Returns

Syscall return tag containing one of the following return codes.

Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<i>thread</i> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	<i>thread</i> is missing L4_CAP_FPAGE_S right.

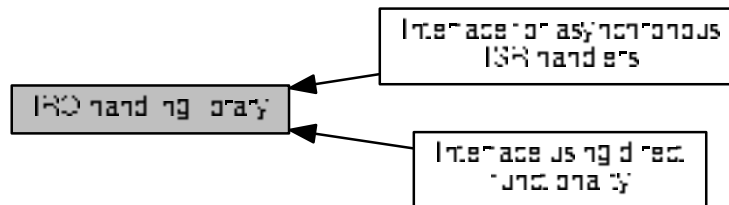
Examples:

[examples/sys/isr/main.c](#).

Definition at line 80 of file [rcv_endpoint.h](#).

12.41 IRQ handling library

Collaboration diagram for IRQ handling library:



Modules

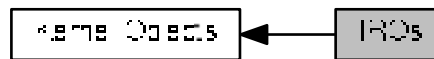
- [Interface for asynchronous ISR handlers.](#)
This interface has just two (main) functions.
- [Interface using direct functionality.](#)

12.41.1 Detailed Description

12.42 IRQs

C IRQ interface.

Collaboration diagram for IRQs:



Enumerations

- enum `L4_irq_mode` {
`L4_IRQ_F_NONE` = 0, `L4_IRQ_F_LEVEL` = 0x2, `L4_IRQ_F_EDGE` = 0x0, `L4_IRQ_F_POS` = 0x0,
`L4_IRQ_F_NEG` = 0x4, `L4_IRQ_F_BOTH` = 0x8, `L4_IRQ_F_LEVEL_HIGH` = 0x3, `L4_IRQ_F_LEVEL_LOW`
= 0x7,
`L4_IRQ_F_POS_EDGE` = 0x1, `L4_IRQ_F_NEG_EDGE` = 0x5, `L4_IRQ_F_BOTH_EDGE` = 0x9, `L4_IRQ_F_MASK` = 0xf,
`L4_IRQ_F_SET_WAKEUP` = 0x10, `L4_IRQ_F_CLEAR_WAKEUP` = 0x20 }

Interrupt attributes.

Functions

- `l4_msgtag_t l4_irq_attach (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread) L4_NOTHROW`
Attach a thread to an interrupt source.
- `l4_msgtag_t l4_irq_attach_u (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW`
Attach a thread to this interrupt.
- `l4_msgtag_t l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW`
Chain an IRQ to another master IRQ source.
- `l4_msgtag_t l4_irq_mux_chain_u (l4_cap_idx_t irq, l4_cap_idx_t slave, l4_utcb_t *utcb) L4_NOTHROW`
Attach an IRQ to this multiplexer.
- `l4_msgtag_t l4_irq_detach (l4_cap_idx_t irq) L4_NOTHROW`
Detach from an interrupt source.
- `l4_msgtag_t l4_irq_detach_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`
Detach from this interrupt.
- `l4_msgtag_t l4_irq_trigger (l4_cap_idx_t irq) L4_NOTHROW`
Trigger an IRQ.
- `l4_msgtag_t l4_irq_trigger_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`
Trigger.
- `l4_msgtag_t l4_irq_receive (l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW`
Unmask and wait for specified IRQ.
- `l4_msgtag_t l4_irq_receive_u (l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW`
Unmask and wait for this IRQ.
- `l4_msgtag_t l4_irq_wait (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW`

Unmask IRQ and wait for any message.

- `l4_msgtag_t l4_irq_wait_u (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t timeout, l4_utcb_t *utcb) L4↔_NOTHROW`

Unmask IRQ and (open) wait for any message.

- `l4_msgtag_t l4_irq_unmask (l4_cap_idx_t irq) L4_NOTHROW`

Unmask IRQ.

- `l4_msgtag_t l4_irq_unmask_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`

Unmask IRQ.

12.42.1 Detailed Description

C IRQ interface.

The IRQ interface provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs)

IRQ objects can be created using a factory, see the [Factory](#) API (use `l4_factory_create_irq()`).

Include File

```
#include <l4/sys/irq.h>
```

For the C++ interface refer to the [L4::Irq](#) API for an overview.

12.42.2 Enumeration Type Documentation

12.42.2.1 L4_irq_mode

```
enum L4_irq_mode
```

Interrupt attributes.

Enumerator

L4_IRQ_F_NONE	Flow types. None
L4_IRQ_F_LEVEL	Level triggered.
L4_IRQ_F_EDGE	Edge triggered.
L4_IRQ_F_POS	Positive trigger.
L4_IRQ_F_NEG	Negative trigger.
L4_IRQ_F_BOTH	Both edges trigger.
L4_IRQ_F_LEVEL_HIGH	Level high trigger.
L4_IRQ_F_LEVEL_LOW	Level low trigger.
L4_IRQ_F_POS_EDGE	Positive edge trigger.
L4_IRQ_F_NEG_EDGE	Negative edge trigger.
L4_IRQ_F_BOTH_EDGE	Both edges trigger.
L4_IRQ_F_MASK	Mask.

Definition at line 67 of file [icu.h](#).

12.42.3 Function Documentation

12.42.3.1 l4_irq_attach()

```
l4_msgtag_t l4_irq_attach (
    l4_cap_idx_t irq,
    l4_umword_t label,
    l4_cap_idx_t thread ) [inline]
```

Attach a thread to an interrupt source.

Parameters

<i>irq</i>	IRQ object where <i>thread</i> is attached to.
<i>label</i>	Identifier of the IRQ.
<i>thread</i>	The thread object to attach <i>irq</i> to.

Returns

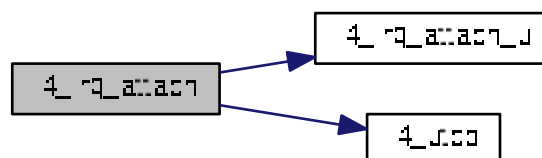
Syscall return tag

The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Definition at line 331 of file [irq.h](#).

References [l4_irq_attach_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.42.3.2 l4_irq_attach_u()

```
l4_msgtag_t l4_irq_attach_u (
    l4_cap_idx_t irq,
    l4_umword_t label,
    l4_cap_idx_t thread,
    l4_utcb_t * utcb ) [inline]
```

Attach a thread to this interrupt.

Parameters

<i>irq</i>	IRQ object where <i>thread</i> is attached to.
<i>label</i>	Identifier of the IRQ (<i>protected label</i> used for messages)
<i>thread</i>	Capability of the thread to attach the IRQ to.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

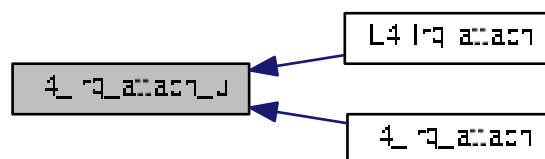
The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Deprecated Use `bind_thread()`.

Definition at line 261 of file `irq.h`.

Referenced by `L4::Irq::attach()`, and `l4_irq_attach()`.

Here is the caller graph for this function:



12.42.3.3 l4_irq_detach()

```
l4_msgtag_t l4_irq_detach (
    l4_cap_idx_t irq ) [inline]
```

Detach from an interrupt source.

Parameters

<i>irq</i>	The IRQ object that shall be detached.
------------	--

Returns

Syscall return tag

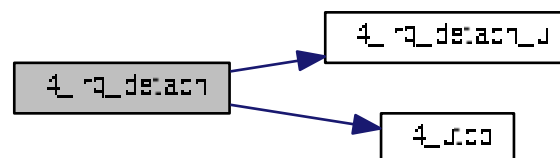
Examples:

[examples/sys/isr/main.c](#).

Definition at line 347 of file [irq.h](#).

References [l4_irq_detach_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**12.42.3.4 l4_irq_detach_u()**

```

l4_msgtag_t l4_irq_detach_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Detach from this interrupt.

Parameters

<i>irq</i>	The IRQ object that shall be detached.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

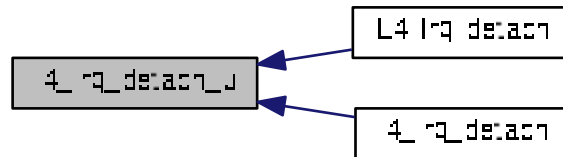
Returns

Syscall return tag

Definition at line 292 of file [irq.h](#).

Referenced by [L4::Irq::detach\(\)](#), and [l4_irq_detach\(\)](#).

Here is the caller graph for this function:



12.42.3.5 l4_irq_mux_chain()

```
l4_msgtag_t l4_irq_mux_chain (
    l4_cap_idx_t irq,
    l4_cap_idx_t slave ) [inline]
```

Chain an IRQ to another master IRQ source.

Parameters

<i>irq</i>	The master IRQ object.
<i>slave</i>	The slave that shall be attached to the master.

Returns

Syscall return tag

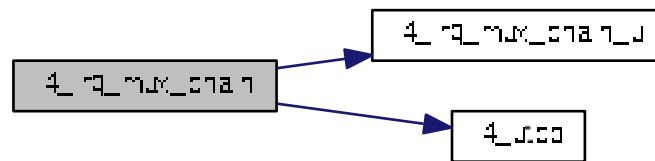
The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread attached to it.

This function allows to add a limited number of slave IRQs to this master IRQ, with the semantics that each of the slave IRQs is triggered whenever the master IRQ is triggered. The master IRQ will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

Definition at line 341 of file [irq.h](#).

References [l4_irq_mux_chain_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.42.3.6 l4_irq_mux_chain_u()

```
l4_msgtag_t l4_irq_mux_chain_u (
    l4_cap_idx_t irq,
    l4_cap_idx_t slave,
    l4_utcb_t * utcb ) [inline]
```

Attach an IRQ to this multiplexer.

Parameters

<i>irq</i>	The master IRQ object.
<i>slave</i>	The slave that shall be attached to the master.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

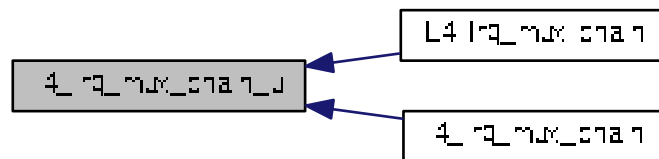
Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer (`Irq_mux`) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 280 of file [irq.h](#).

Referenced by [L4::Irq_mux::chain\(\)](#), and [l4_irq_mux_chain\(\)](#).

Here is the caller graph for this function:



12.42.3.7 l4_irq_receive()

```
l4_msgtag_t l4_irq_receive (
    l4_cap_idx_t irq,
    l4_timeout_t to ) [inline]
```

Unmask and wait for specified IRQ.

Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>to</i>	Timeout.

Returns

Syscall return tag

Examples:

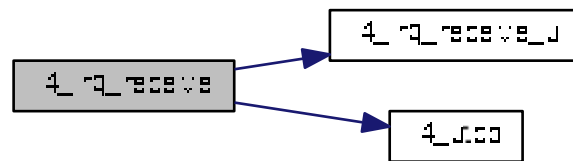
[examples/sys/isr/main.c](#).

Definition at line 359 of file [irq.h](#).

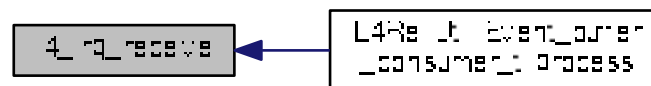
References [l4_irq_receive_u\(\)](#), and [l4_utcb\(\)](#).

Referenced by [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.42.3.8 l4_irq_receive_u()

```

l4_msgtag_t l4_irq_receive_u (
    l4_cap_idx_t irq,
    l4_timeout_t timeout,
    l4_utcb_t * utcb ) [inline]
  
```

Unmask and wait for this IRQ.

Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

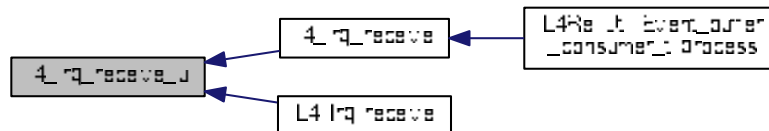
Note

If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 307 of file [irq.h](#).

Referenced by [l4_irq_receive\(\)](#), and [L4::Irq::receive\(\)](#).

Here is the caller graph for this function:

**12.42.3.9 l4_irq_trigger()**

```
l4_msgtag_t l4_irq_trigger (
    l4_cap_idx_t irq ) [inline]
```

Trigger an IRQ.

Parameters

<i>irq</i>	The IRQ object that shall be triggered.
------------	---

Returns

Syscall return tag.

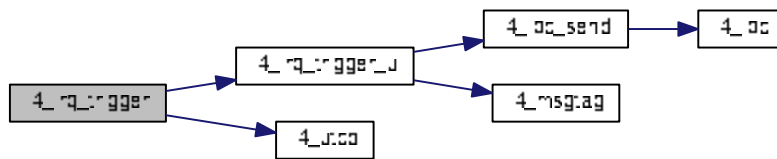
Note that this function is a send only operation, i.e. there is no return value except for a failed send operation. Especially [l4_error\(\)](#) will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

Use [l4_ipc_error\(\)](#) to check for (send) errors.

Definition at line 353 of file [irq.h](#).

References [l4_irq_trigger_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.42.3.10 l4_irq_trigger_u()

```

l4_msgtag_t l4_irq_trigger_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Trigger.

Parameters

<i>irq</i>	The IRQ object that shall be triggered.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

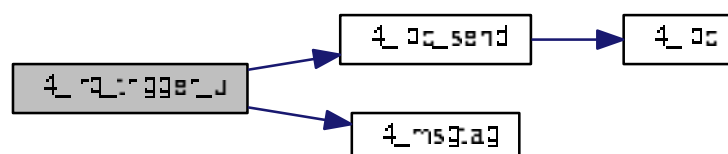
This function is a send-only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Definition at line 300 of file [irq.h](#).

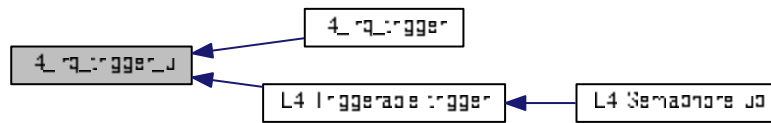
References [L4_IPC_BOTH_TIMEOUT_0](#), [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), and [L4_PROTO_IRQ](#).

Referenced by [l4_irq_trigger\(\)](#), and [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.42.3.11 `l4_irq_unmask()`

```
l4_msgtag_t l4_irq_unmask (
    l4_cap_idx_t irq ) [inline]
```

Unmask IRQ.

Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
------------	--

Returns

Syscall return tag

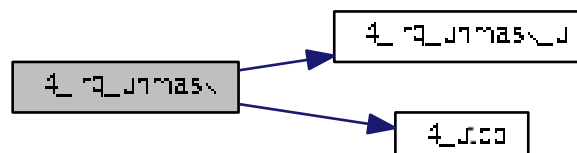
Note

`l4_irq_wait()` and `l4_irq_receive()` are doing the unmask themselves.

Definition at line 372 of file irq.h.

References `l4_irq_unmask_u()`, and `l4_utcb()`.

Here is the call graph for this function:



12.42.3.12 l4_irq_unmask_u()

```
l4_msgtag_t l4_irq_unmask_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
```

Unmask IRQ.

Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

This function is a send-only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

`Irq::wait()` and `Irq::receive()` operations already include an `unmask()`, do not use an extra `unmask()` in these cases.

Deprecated Use `L4::Irq_eoi::unmask()`

Definition at line 323 of file [irq.h](#).

Referenced by [l4_irq_unmask\(\)](#).

Here is the caller graph for this function:



12.42.3.13 l4_irq_wait()

```
l4_msgtag_t l4_irq_wait (
    l4_cap_idx_t irq,
    l4_umword_t * label,
    l4_timeout_t to ) [inline]
```

Unmask IRQ and wait for any message.

Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>label</i>	Receive label.
<i>to</i>	Timeout.

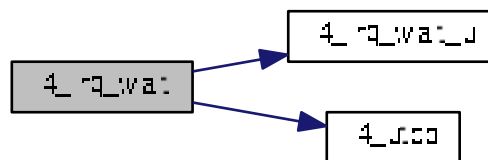
Returns

Syscall return tag

Definition at line 365 of file [irq.h](#).

References [l4_irq_wait_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.42.3.14 l4_irq_wait_u()

```

l4_msgtag_t l4_irq_wait_u (
    l4_cap_idx_t irq,
    l4_umword_t * label,
    l4_timeout_t timeout,
    l4_utcb_t * utcb ) [inline]
  
```

Unmask IRQ and (open) wait for any message.

Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>label</i>	The <i>protected label</i> shall be received here.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 314 of file [irq.h](#).

Referenced by [l4_irq_wait\(\)](#).

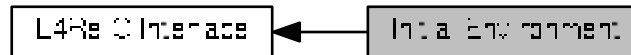
Here is the caller graph for this function:



12.43 Initial Environment

C interface of the initial environment that is provided to an [L4](#) task.

Collaboration diagram for Initial Environment:



Data Structures

- struct [l4re_env_cap_entry_t](#)
Entry in the [L4Re](#) environment array for the named initial objects.

Typedefs

- typedef struct [l4re_env_cap_entry_t](#) [l4re_env_cap_entry_t](#)
Entry in the [L4Re](#) environment array for the named initial objects.

Functions

- [l4re_env_t * l4re_env](#) (void) [L4_NOTHROW](#)
Get [L4Re](#) initial environment.
- [l4_kernel_info_t * l4re_kip](#) (void) [L4_NOTHROW](#)
Get Kernel Info Page.
- [l4_cap_idx_t l4re_env_get_cap](#) (char const *name) [L4_NOTHROW](#)
Get the capability selector for the object named name.
- [l4_cap_idx_t l4re_env_get_cap_e](#) (char const *name, [l4re_env_t](#) const *e) [L4_NOTHROW](#)
Get the capability selector for the object named name.
- [l4re_env_cap_entry_t](#) const * [l4re_env_get_cap_l](#) (char const *name, unsigned l, [l4re_env_t](#) const *e) [L4_NOTHROW](#)
Get the full [l4re_env_cap_entry_t](#) for the object named name.

12.43.1 Detailed Description

C interface of the initial environment that is provided to an [L4](#) task.

Include File

```
#include <l4/re/env.h>
```

For an explanation of the default task capabilities see [l4_default_caps_t](#).

For the C++ interface refer to [L4Re::Env](#).

12.43.2 Function Documentation

12.43.2.1 `l4re_env()`

```
l4re_env_t * l4re_env (
    void ) [inline]
```

Get [L4Re](#) initial environment.

Returns

Pointer to [L4Re](#) initial environment.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [184](#) of file [env.h](#).

Referenced by [l4re_env_get_cap\(\)](#).

Here is the caller graph for this function:



12.43.2.2 `l4re_env_get_cap()`

```
l4_cap_idx_t l4re_env_get_cap (
    char const * name ) [inline]
```

Get the capability selector for the object named *name*.

Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
-------------	---

Returns

A valid capability selector if the object exists or an invalid capability selector if not ([l4_is_invalid_cap\(\)](#)).

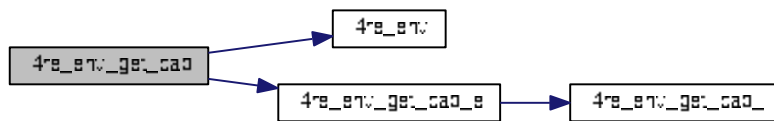
Examples:

[examples/sys/isr/main.c](#).

Definition at line 226 of file [env.h](#).

References [l4re_env\(\)](#), [l4re_env_get_cap_e\(\)](#), and [l4re_env_cap_entry_t::name](#).

Here is the call graph for this function:

**12.43.2.3 l4re_env_get_cap_e()**

```

l4_cap_idx_t l4re_env_get_cap_e (
    char const * name,
    l4re_env_t const * e ) [inline]
  
```

Get the capability selector for the object named *name*.

Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
<i>e</i>	is the environment structure to use for the operation.

Returns

A valid capability selector if the object exists or an invalid capability selector if not ([l4_is_invalid_cap\(\)](#)).

Definition at line 213 of file [env.h](#).

References [l4re_env_cap_entry_t::cap](#), [L4_INVALID_CAP](#), [l4re_env_get_cap_l\(\)](#), and [l4re_env_cap_entry_t::name](#).

Referenced by [l4re_env_get_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.43.2.4 l4re_env_get_cap_l()

```

l4re_env_cap_entry_t const * l4re_env_get_cap_l (
    char const * name,
    unsigned l,
    l4re_env_t const * e ) [inline]
  
```

Get the full [l4re_env_cap_entry_t](#) for the object named *name*.

Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
<i>l</i>	is the length of the name string, thus <i>name</i> might not be zero terminated.
<i>e</i>	is the environment structure to use for the operation.

Returns

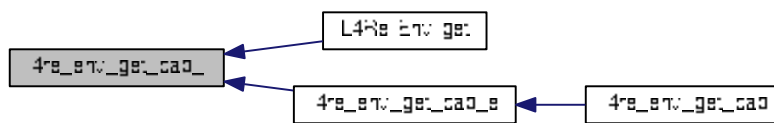
A pointer to an [l4re_env_cap_entry_t](#) if the object exists or NULL if not.

Definition at line 195 of file [env.h](#).

References [l4re_env_cap_entry_t::flags](#), and [l4re_env_cap_entry_t::name](#).

Referenced by [L4Re::Env::get\(\)](#), and [l4re_env_get_cap_e\(\)](#).

Here is the caller graph for this function:



12.43.2.5 l4re_kip()

```
l4_kernel_info_t * l4re_kip (
    void ) [inline]
```

Get Kernel Info Page.

Returns

Pointer to Kernel Info Page (KIP) structure.

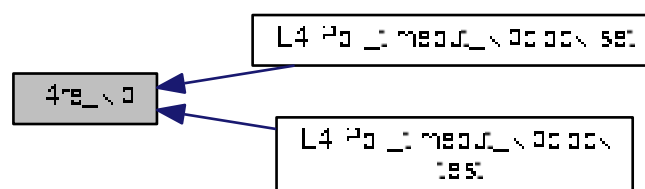
Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/ux-vhw/main.c](#).

Definition at line 188 of file [env.h](#).

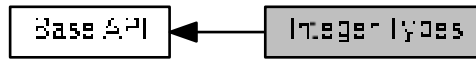
Referenced by [L4::Poll_timeout_kipclock::set\(\)](#), and [L4::Poll_timeout_kipclock::test\(\)](#).

Here is the caller graph for this function:



12.44 Integer Types

Collaboration diagram for Integer Types:



Files

- file [l4int.h](#)
Fixed sized integer types, generic version.
- file [l4int.h](#)
Fixed sized integer types, arm version.
- file [l4int.h](#)
Fixed sized integer types, amd64 version.
- file [l4int.h](#)
Fixed sized integer types, x86 version.

Macros

- `#define L4_MWORD_BITS 32`
Size of machine words in bits.
- `#define L4_MWORD_BITS 64`
Size of machine words in bits.
- `#define L4_MWORD_BITS 32`
Size of machine words in bits.

Typedefs

- typedef signed char [l4_int8_t](#)
Signed 8bit value.
- typedef unsigned char [l4_uint8_t](#)
Unsigned 8bit value.
- typedef signed short int [l4_int16_t](#)
Signed 16bit value.
- typedef unsigned short int [l4_uint16_t](#)
Unsigned 16bit value.
- typedef signed int [l4_int32_t](#)
Signed 32bit value.
- typedef unsigned int [l4_uint32_t](#)
Unsigned 32bit value.
- typedef signed long long [l4_int64_t](#)

Signed 64bit value.

- typedef unsigned long long [l4_uint64_t](#)

Unsigned 64bit value.

- typedef unsigned long [l4_addr_t](#)

Address type.

- typedef signed long [l4_mword_t](#)

Signed machine word.

- typedef unsigned long [l4_umword_t](#)

Unsigned machine word.

- typedef [l4_uint64_t](#) [l4_cpu_time_t](#)

CPU clock type.

- typedef [l4_uint64_t](#) [l4_kernel_clock_t](#)

Kernel clock type.

- typedef unsigned int [l4_size_t](#)

Unsigned size type.

- typedef signed int [l4_ssize_t](#)

Signed size type.

- typedef unsigned long [l4_size_t](#)

Unsigned size type.

- typedef signed long [l4_ssize_t](#)

Signed size type.

- typedef unsigned int [l4_size_t](#)

Unsigned size type.

- typedef signed int [l4_ssize_t](#)

Signed size type.

12.44.1 Detailed Description

Include File

```
#include <l4/sys/l4int.h>
```

12.44.2 Typedef Documentation

12.44.2.1 l4_int16_t

```
typedef signed short int l4\_int16\_t
```

Signed 16bit value.

Definition at line 37 of file [l4int.h](#).

12.44.2.2 l4_int32_t

```
typedef signed int l4_int32_t
```

Signed 32bit value.

Definition at line 39 of file [l4int.h](#).

12.44.2.3 l4_int64_t

```
typedef signed long long l4_int64_t
```

Signed 64bit value.

Definition at line 41 of file [l4int.h](#).

12.44.2.4 l4_int8_t

```
typedef signed char l4_int8_t
```

Signed 8bit value.

Definition at line 35 of file [l4int.h](#).

12.44.2.5 l4_uint16_t

```
typedef unsigned short int l4_uint16_t
```

Unsigned 16bit value.

Definition at line 38 of file [l4int.h](#).

12.44.2.6 l4_uint32_t

```
typedef unsigned int l4_uint32_t
```

Unsigned 32bit value.

Definition at line 40 of file [l4int.h](#).

12.44.2.7 l4_uint64_t

```
typedef unsigned long long l4_uint64_t
```

Unsigned 64bit value.

Definition at line 42 of file [l4int.h](#).

12.44.2.8 l4_uint8_t

```
typedef unsigned char l4_uint8_t
```

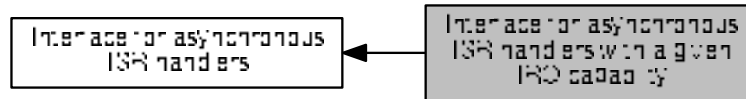
Unsigned 8bit value.

Definition at line 36 of file [l4int.h](#).

12.45 Interface for asynchronous ISR handlers with a given IRQ capability.

This group is just an enhanced version to [l4irq_request\(\)](#) which takes a capability object instead of a plain number.

Collaboration diagram for Interface for asynchronous ISR handlers with a given IRQ capability.:



Functions

- `l4irq_t * l4irq_request_cap (l4_cap_idx_t irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)`

Attach asynchronous ISR handler to IRQ.

12.45.1 Detailed Description

This group is just an enhanced version to [l4irq_request\(\)](#) which takes a capability object instead of a plain number.

12.45.2 Function Documentation

12.45.2.1 l4irq_request_cap()

```

l4irq_t* l4irq_request_cap (
    l4_cap_idx_t irqcap,
    void(*) (void *) isr_handler,
    void * isr_data,
    int irq_thread_prio,
    unsigned mode )
  
```

Attach asynchronous ISR handler to IRQ.

Parameters

<i>irqcap</i>	IRQ capability
<i>isr_handler</i>	Handler routine that is called when an interrupt triggers
<i>isr_data</i>	Pointer given as argument to <i>isr_handler</i>
<i>irq_thread_prio</i>	L4 thread priority of the ISR handler. Give -1 for same priority as creator.
<i>mode</i>	Interrupt type,

See also

[L4_irq_mode](#)

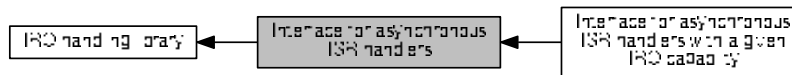
Returns

Pointer to `l4irq_t` structure, 0 on error

12.46 Interface for asynchronous ISR handlers.

This interface has just two (main) functions.

Collaboration diagram for Interface for asynchronous ISR handlers.:



Modules

- [Interface for asynchronous ISR handlers with a given IRQ capability.](#)

This group is just an enhanced version to [l4irq_request\(\)](#) which takes a capability object instead of a plain number.

Functions

- `l4irq_t * l4irq_request (int irqnum, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)`
Attach asynchronous ISR handler to IRQ.
- `long l4irq_release (l4irq_t *irq)`
Release asynchronous ISR handler and free resources.

12.46.1 Detailed Description

This interface has just two (main) functions.

`l4irq_request` to install a handler for an interrupt and `l4irq_release` to uninstall the handler again and release all resources associated with it.

12.46.2 Function Documentation

12.46.2.1 `l4irq_release()`

```
long l4irq_release (
    l4irq_t * irq )
```

Release asynchronous ISR handler and free resources.

Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

Returns

0 success, != 0 failure

Examples:

[examples/libs/libirq/async_isr.c](#).

12.46.2.2 l4irq_request()

```
l4irq_t* l4irq_request (
    int irqnum,
    void(*) (void *) isr_handler,
    void * isr_data,
    int irq_thread_prio,
    unsigned mode )
```

Attach asynchronous ISR handler to IRQ.

Parameters

<i>irqnum</i>	IRQ number to request
<i>isr_handler</i>	Handler routine that is called when an interrupt triggers
<i>isr_data</i>	Pointer given as argument to isr_handler
<i>irq_thread_prio</i>	L4 thread priority of the ISR handler. Give -1 for same priority as creator.
<i>mode</i>	Interrupt type,

See also

[L4_irq_mode](#)

Returns

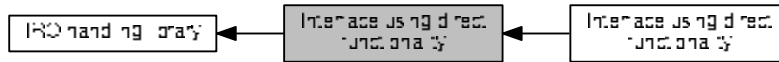
Pointer to l4irq_t structure, 0 on error

Examples:

[examples/libs/libirq/async_isr.c](#).

12.47 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



Modules

- [Interface using direct functionality.](#)

Functions

- `l4irq_t * l4irq_attach (int irqnum)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_ft (int irqnum, unsigned mode)`
Attach/connect to IRQ using given type.
- `l4irq_t * l4irq_attach_thread (int irqnum, l4_cap_idx_t to_thread)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_thread_ft (int irqnum, l4_cap_idx_t to_thread, unsigned mode)`
Attach/connect to IRQ using given type.
- `long l4irq_wait (l4irq_t *irq)`
Wait for specified IRQ.
- `long l4irq_unmask_and_wait_any (l4irq_t *unmask_irq, l4irq_t **ret_irq)`
Unmask a specific IRQ and wait for any attached IRQ.
- `long l4irq_wait_any (l4irq_t **irq)`
Wait for any attached IRQ.
- `long l4irq_unmask (l4irq_t *irq)`
Unmask a specific IRQ.
- `long l4irq_detach (l4irq_t *irq)`
Detach from IRQ.

12.47.1 Detailed Description

12.47.2 Function Documentation

12.47.2.1 l4irq_attach()

```
l4irq_t* l4irq_attach (
    int irqnum )
```

Attach/connect to IRQ.

Parameters

<i>irqnum</i>	IRQ number to request
---------------	-----------------------

Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

Examples:

[examples/libs/libirq/loop.c](#).

12.47.2.2 l4irq_attach_ft()

```
l4irq_t* l4irq_attach_ft (
    int irqnum,
    unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

<i>irqnum</i>	IRQ number to request
<i>mode</i>	Interrupt type,

See also

[L4_irq_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

12.47.2.3 l4irq_attach_thread()

```
l4irq_t* l4irq_attach_thread (
    int irqnum,
    l4_cap_idx_t to_thread )
```

Attach/connect to IRQ.

Parameters

<i>irqnum</i>	IRQ number to request
<i>to_thread</i>	Attach IRQ to this specified thread.

Returns

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

12.47.2.4 l4irq_attach_thread_ft()

```
l4irq_t* l4irq_attach_thread_ft (
    int irqnum,
    l4_cap_idx_t to_thread,
    unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

<i>irqnum</i>	IRQ number to request
<i>to_thread</i>	Attach IRQ to this specified thread.
<i>mode</i>	Interrupt type,

See also

[L4_irq_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

12.47.2.5 l4irq_detach()

```
long l4irq_detach (
    l4irq_t * irq )
```

Detach from IRQ.

Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

Returns

0 on success, != 0 on error

12.47.2.6 l4irq_unmask()

```
long l4irq_unmask (
    l4irq_t * irq )
```

Unmask a specific IRQ.

Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

Returns

0 on success, != 0 on error

This function is useful if a thread wants to wait for multiple IRQs using l4_ipc_wait.

12.47.2.7 l4irq_unmask_and_wait_any()

```
long l4irq_unmask_and_wait_any (
    l4irq_t * unmask_irq,
    l4irq_t ** ret_irq )
```

Unmask a specific IRQ and wait for any attached IRQ.

Parameters

<i>unmask_irq</i>	IRQ data structure for unmask.
-------------------	--------------------------------

Return values

<i>ret_irq</i>	Received interrupt.
----------------	---------------------

Returns

0 on success, != 0 on error

12.47.2.8 l4irq_wait()

```
long l4irq_wait (
    l4irq_t * irq )
```


Wait for specified IRQ.

Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

Returns

0 on success, != 0 on error

Examples:

[examples/libs/libirq/loop.c](#).

12.47.2.9 l4irq_wait_any()

```
long l4irq_wait_any (
    l4irq_t ** irq )
```

Wait for any attached IRQ.

Return values

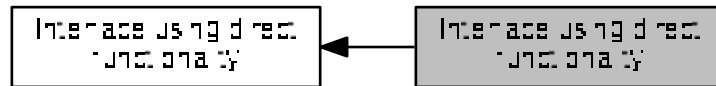
<i>irq</i>	Received interrupt.
------------	---------------------

Returns

0 on success, != 0 on error

12.48 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



Functions

- `l4irq_t * l4irq_attach_cap (l4_cap_idx_t irqcap)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_cap_ft (l4_cap_idx_t irqcap, unsigned mode)`
Attach/connect to IRQ using given type.
- `l4irq_t * l4irq_attach_thread_cap (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)`
Attach/connect to IRQ using given type.

12.48.1 Detailed Description

12.48.2 Function Documentation

12.48.2.1 `l4irq_attach_cap()`

```
l4irq_t* l4irq_attach_cap (
    l4_cap_idx_t irqcap )
```

Attach/connect to IRQ.

Parameters

<code>irqcap</code>	IRQ capability
---------------------	----------------

Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

12.48.2.2 l4irq_attach_cap_ft()

```
l4irq_t* l4irq_attach_cap_ft (
    l4_cap_idx_t irqcap,
    unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

<i>irqcap</i>	IRQ capability
<i>mode</i>	Interrupt type,

See also

[L4_irq_mode](#)

Returns

Pointer to l4irq_t structure, 0 on error

This l4irq_attach has to be called in the same thread as l4irq_wait and caller has to be a pthread thread.

12.48.2.3 l4irq_attach_thread_cap()

```
l4irq_t* l4irq_attach_thread_cap (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread )
```

Attach/connect to IRQ.

Parameters

<i>irqcap</i>	IRQ capability
<i>to_thread</i>	Attach IRQ to this thread.

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

12.48.2.4 l4irq_attach_thread_cap_ft()

```
l4irq_t* l4irq_attach_thread_cap_ft (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread,
    unsigned mode )
```

Attach/connect to IRQ using given type.

Parameters

<i>irqcap</i>	IRQ capability
<i>to_thread</i>	Attach IRQ to this thread.
<i>mode</i>	Interrupt type,

See also

[L4_irq_mode](#)

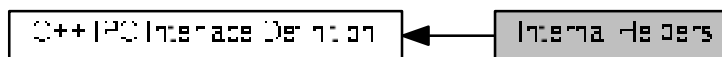
Returns

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

12.49 Internal Helpers

Collaboration diagram for Internal Helpers:



Data Structures

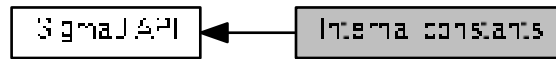
- struct [L4::Types::Bool< V >](#)
Boolean meta type.
- struct [L4::Types::False](#)
False meta value.
- struct [L4::Types::True](#)
True meta value.
- struct [L4::Types::Same< A, B >](#)
Compare two data types for equality.

12.49.1 Detailed Description

12.50 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:



Macros

- #define `SIGMA0_REQ_MAGIC` ~0xFFUL
Request magic.
- #define `SIGMA0_REQ_MASK` ~0xFFUL
Request mask.
- #define `SIGMA0_REQ_ID_MASK` 0xF0
ID mask.
- #define `SIGMA0_REQ_ID_FPAGE_RAM` 0x60
RAM.
- #define `SIGMA0_REQ_ID_FPAGE_IOMEM` 0x70
I/O memory.
- #define `SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED` 0x80
Cached I/O memory.
- #define `SIGMA0_REQ_ID_FPAGE_ANY` 0x90
Any.
- #define `SIGMA0_REQ_ID_KIP` 0xA0
KIP.
- #define `SIGMA0_REQ_ID_TBUF` 0xB0
TBUF.
- #define `SIGMA0_REQ_ID_DEBUG_DUMP` 0xC0
Debug dump.
- #define `SIGMA0_REQ_ID_NEW_CLIENT` 0xD0
New client.
- #define `SIGMA0_IS_MAGIC_REQ(d1)` ((d1 & `SIGMA0_REQ_MASK`) == `SIGMA0_REQ_MAGIC`)
Check if magic.
- #define `SIGMA0_REQ(x)` (`SIGMA0_REQ_MAGIC` + `SIGMA0_REQ_ID_## x`)
Construct.
- #define `SIGMA0_REQ_FPAGE_RAM` (`SIGMA0_REQ`(`FPAGE_RAM`))
RAM.
- #define `SIGMA0_REQ_FPAGE_IOMEM` (`SIGMA0_REQ`(`FPAGE_IOMEM`))
I/O memory.
- #define `SIGMA0_REQ_FPAGE_IOMEM_CACHED` (`SIGMA0_REQ`(`FPAGE_IOMEM_CACHED`))
Cache I/O memory.
- #define `SIGMA0_REQ_FPAGE_ANY` (`SIGMA0_REQ`(`FPAGE_ANY`))

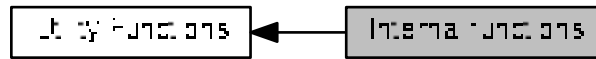
- Any.*
 - #define `SIGMA0_REQ_KIP` (`SIGMA0_REQ(KIP)`)
- KIP.*
 - #define `SIGMA0_REQ_TBUF` (`SIGMA0_REQ(TBUF)`)
- TBUF.*
 - #define `SIGMA0_REQ_DEBUG_DUMP` (`SIGMA0_REQ(DEBUG_DUMP)`)
- Debug dump.*
 - #define `SIGMA0_REQ_NEW_CLIENT` (`SIGMA0_REQ(NEW_CLIENT)`)
- New client.*

12.50.1 Detailed Description

Internal sigma0 definitions.

12.51 Internal functions

Collaboration diagram for Internal functions:



Functions

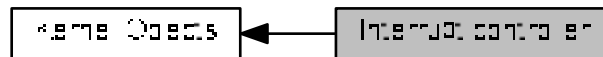
- void [base64_encode](#) (const char *infile, unsigned int in_size, char **outfile)
base-64-encode string infile
- void [base64_decode](#) (const char *infile, unsigned int in_size, char **outfile)
decode base-64-encoded string infile

12.51.1 Detailed Description

12.52 Interrupt controller

The C Icu interface.

Collaboration diagram for Interrupt controller:



Data Structures

- struct [l4_icu_info_t](#)
Info structure for an ICU.

Typedefs

- typedef struct [l4_icu_info_t](#) [l4_icu_info_t](#)
Info structure for an ICU.

Enumerations

- enum [L4_icu_flags](#) { [L4_ICU_FLAG_MSI](#) }
Flags for IRQ numbers used for the ICU.

Functions

- [l4_msgtag_t l4_icu_bind](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t l4_icu_bind_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t l4_icu_unbind](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t l4_icu_unbind_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t l4_icu_set_mode](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) mode) [L4_NOTHROW](#)
Set interrupt mode.
- [l4_msgtag_t l4_icu_set_mode_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Set interrupt mode.
- [l4_msgtag_t l4_icu_info](#) ([l4_cap_idx_t](#) icu, [l4_icu_info_t](#) *info) [L4_NOTHROW](#)

Get information about the capabilities of the ICU.

- `l4_msgtag_t l4_icu_info_u (l4_cap_idx_t icu, l4_icu_info_t *info, l4_utcb_t *utcb) L4_NOTHROW`

Get information about the capabilities of the ICU.

- `l4_msgtag_t l4_icu_msi_info (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info) L4_NOTHROW`

Get MSI info about IRQ.

- `l4_msgtag_t l4_icu_msi_info_u (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW`

Get MSI info about IRQ.

- `l4_msgtag_t l4_icu_unmask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW`

Unmask an IRQ line.

- `l4_msgtag_t l4_icu_unmask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW`

Acknowledge the given interrupt line.

- `l4_msgtag_t l4_icu_mask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW`

Mask an IRQ line.

- `l4_msgtag_t l4_icu_mask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW`

Mask an IRQ line.

12.52.1 Detailed Description

The C Icu interface.

To setup an IRQ line the following steps are required:

1. `l4_icu_set_mode()` (optional if IRQ has a default mode)
2. `l4_irq_attach()` to attach the IRQ capability to a thread
3. `l4_icu_bind()`
4. `l4_icu_unmask()` to receive the first IRQ

Include File

```
#include <l4/sys/icu.h>
```

12.52.2 Typedef Documentation

12.52.2.1 l4_icu_info_t

```
typedef struct l4_icu_info_t l4_icu_info_t
```

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

`l4_icu_info()`.

12.52.3 Enumeration Type Documentation

12.52.3.1 L4_icu_flags

enum [L4_icu_flags](#)

Flags for IRQ numbers used for the ICU.

Enumerator

L4_ICU_FLAG_MSI	Flag to denote that the IRQ is actually an MSI. This flag may be used for l4_icu_bind() and l4_icu_unbind() functions to denote that the IRQ number is meant to be an MSI.
---------------------------------	--

Definition at line 50 of file [icu.h](#).

12.52.4 Function Documentation

12.52.4.1 l4_icu_bind()

```
l4_msgtag_t l4_icu_bind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq ) [inline]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

<i>icu</i>	ICU object to bind <i>irq</i> to.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to bind to this ICU.

Returns

Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [l4_irq_unmask\(\)](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [l4_icu_unmask\(\)](#).

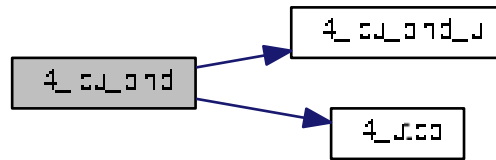
Examples:

[examples/sys/isr/main.c](#).

Definition at line 473 of file [icu.h](#).

References [l4_icu_bind_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.52.4.2 l4_icu_bind_u()

```

l4_msgtag_t l4_icu_bind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

<i>icu</i>	The ICU object to bind <i>irq</i> to.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object for the given IRQ line to bind to this ICU.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

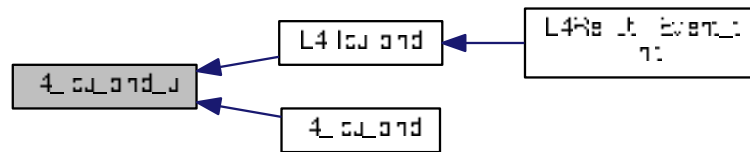
Returns

Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::icu::unmask](#).

Definition at line 373 of file [icu.h](#).

Referenced by [L4::icu::bind\(\)](#), and [l4_icu_bind\(\)](#).

Here is the caller graph for this function:



12.52.4.3 l4_icu_info()

```

l4_msgtag_t l4_icu_info (
    l4_cap_idx_t icu,
    l4_icu_info_t * info ) [inline]
  
```

Get information about the capabilities of the ICU.

Parameters

	<i>icu</i>	The ICU object from which information shall be retrieved.
out	<i>info</i>	Pointer to an info structure to be filled with information. The memory for this structure has to be allocated by the caller.

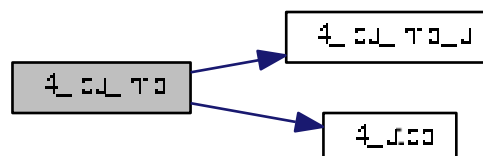
Returns

Syscall return tag

Definition at line 481 of file [icu.h](#).

References [l4_icu_info_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.52.4.4 l4_icu_info_u()

```
l4_msgtag_t l4_icu_info_u (
    l4_cap_idx_t icu,
    l4_icu_info_t * info,
    l4_utcb_t * utcb ) [inline]
```

Get information about the capabilities of the ICU.

Parameters

	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
out	<i>info</i>	Info structure to be filled with information.
	<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

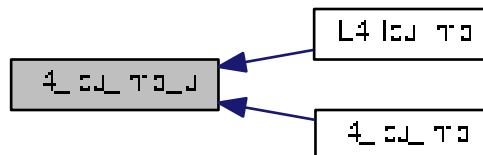
Returns

Syscall return tag

Definition at line 397 of file [icu.h](#).

Referenced by [L4::Icu::info\(\)](#), and [l4_icu_info\(\)](#).

Here is the caller graph for this function:



12.52.4.5 l4_icu_mask()

```
l4_msgtag_t l4_icu_mask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to ) [inline]
```

Mask an IRQ line.

Parameters

<i>icu</i>	The ICU object where the IRQ line 'irqnum' shall be masked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL the function also waits for the next message.
<i>to</i>	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag

Definition at line 495 of file [icu.h](#).

12.52.4.6 l4_icu_mask_u()

```
l4_msgtag_t l4_icu_mask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb ) [inline]
```

Mask an IRQ line.

Parameters

<i>icu</i>	The ICU object where the IRQ line 'irqnum' shall be masked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If NULL this function is a send-only message to the ICU. If not NULL this function will enter an open wait after sending the mask message.
<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 460 of file [icu.h](#).

Referenced by [L4::Icu::mask\(\)](#).

Here is the caller graph for this function:



12.52.4.7 l4_icu_msi_info()

```

l4_msgtag_t l4_icu_msi_info (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info ) [inline]
  
```

Get MSI info about IRQ.

Parameters

	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A <code>l4_icu_msi_info_t</code> structure receiving the address and the data value to trigger this MSI.

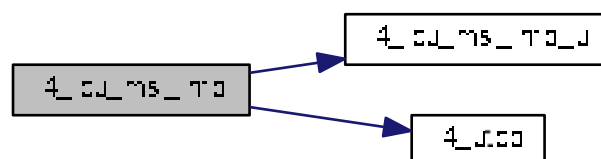
Returns

Syscall return tag

Definition at line 485 of file `icu.h`.

References `l4_icu_msi_info_u()`, and `l4_utcb()`.

Here is the call graph for this function:



12.52.4.8 l4_icu_msi_info_u()

```
l4_msgtag_t l4_icu_msi_info_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info,
    l4_utcb_t * utcb ) [inline]
```

Get MSI info about IRQ.

Parameters

	<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

Returns

Syscall return tag

Definition at line 411 of file [icu.h](#).

Referenced by [l4_icu_msi_info\(\)](#).

Here is the caller graph for this function:



12.52.4.9 l4_icu_set_mode()

```
l4_msgtag_t l4_icu_set_mode (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode ) [inline]
```

Set interrupt mode.

Parameters

<i>icu</i>	The ICU object.
<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see L4_irq_mode .

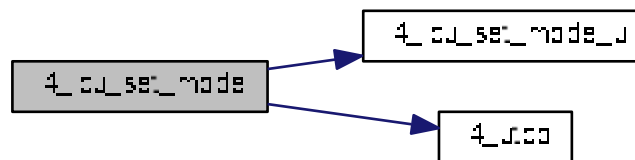
Returns

Syscall return tag

Definition at line 500 of file [icu.h](#).

References [l4_icu_set_mode_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

12.52.4.10 `l4_icu_set_mode_u()`

```

l4_msgtag_t l4_icu_set_mode_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode,
    l4_utcb_t * utcb ) [inline]
  
```

Set interrupt mode.

Parameters

<i>icu</i>	The ICU object.
<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see L4_irq_mode .
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

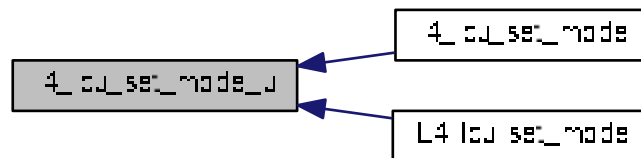
Returns

Syscall return tag

Definition at line 434 of file [icu.h](#).

Referenced by [l4_icu_set_mode\(\)](#), and [L4::Icu::set_mode\(\)](#).

Here is the caller graph for this function:

**12.52.4.11 l4_icu_unbind()**

```

l4_msgtag_t l4_icu_unbind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq ) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

<i>icu</i>	The ICU object from where the binding shall be removed.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.

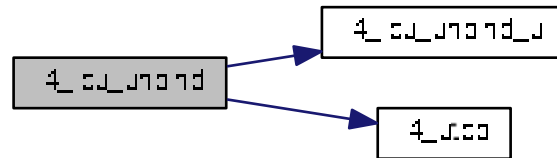
Returns

Syscall return tag

Definition at line 477 of file [icu.h](#).

References [l4_icu_unbind_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.52.4.12 l4_icu_unbind_u()

```

l4_msgtag_t l4_icu_unbind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

<i>icu</i>	The ICU object from where the binding shall be removed.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

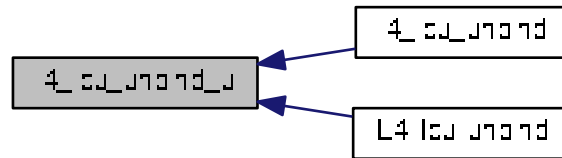
Returns

Syscall return tag

Definition at line 385 of file [icu.h](#).

Referenced by [l4_icu_unbind\(\)](#), and [L4::Icu::unbind\(\)](#).

Here is the caller graph for this function:



12.52.4.13 l4_icu_unmask()

```
l4_msgtag_t l4_icu_unmask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to ) [inline]
```

Unmask an IRQ line.

Parameters

<i>icu</i>	The ICU object where the IRQ line shall be unmasked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL the function also waits for the next message.
<i>to</i>	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag, the error values therein are undefined because `l4_icu_unmask()` is a sender-only IPC.

Definition at line 490 of file `icu.h`.

12.52.4.14 l4_icu_unmask_u()

```
l4_msgtag_t l4_icu_unmask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb ) [inline]
```

Acknowledge the given interrupt line.

Parameters

	<i>icu</i>	The ICU object where the IRQ line shall be unmasked.
	<i>irqnum</i>	The interrupt line that shall be acknowledged.
out	<i>label</i>	If NULL this is a send-only unmask, if not NULL then this operation enters an open wait and the <i>protected label</i> shall be received here.
	<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
	<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag.

Note

If *label* is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Definition at line [465](#) of file [icu.h](#).

12.53 Kernel Debugger

Kernel debugger related functionality.

Collaboration diagram for Kernel Debugger:



Functions

- [l4_msgtag_t l4_debugger_set_object_name \(l4_cap_idx_t cap, const char *name\) L4_NOTHROW](#)
Set the name of a kernel object.
- [unsigned long l4_debugger_global_id \(l4_cap_idx_t cap\) L4_NOTHROW](#)
Get the globally unique ID of the object behind a capability.
- [unsigned long l4_debugger_kobj_to_id \(l4_cap_idx_t cap, l4_addr_t kobjp\) L4_NOTHROW](#)
Get the globally unique ID of the object behind the kobject pointer.

12.53.1 Detailed Description

Kernel debugger related functionality.

Attention

This API is subject to change!

This is a debugging facility, any call to any function might be invalid. Do not rely on it in any real code.

Include File

```
#include <l4/sys/debugger.h>
```

12.53.2 Function Documentation

12.53.2.1 l4_debugger_global_id()

```
unsigned long l4_debugger_global_id (
    l4_cap_idx_t cap ) [inline]
```

Get the globally unique ID of the object behind a capability.

Parameters

<i>cap</i>	Capability
------------	------------

Return values

$\sim 0UL$	Capability is not valid.
≥ 0	Global debugger id.

This is a debugging facility, the call might be invalid.

Definition at line 331 of file [debugger.h](#).

12.53.2.2 l4_debugger_kobj_to_id()

```
unsigned long l4_debugger_kobj_to_id (  
    l4_cap_idx_t cap,  
    l4_addr_t kobjp ) [inline]
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

<i>cap</i>	Capability
<i>kobjp</i>	Kobject pointer

Return values

$\sim 0UL$	The capability or the kobject pointer are invalid.
≥ 0	The globally unique id.

This is a debugging facility, the call might be invalid.

Definition at line 337 of file [debugger.h](#).

12.53.2.3 l4_debugger_set_object_name()

```
l4_msgtag_t l4_debugger_set_object_name (  
    l4_cap_idx_t cap,  
    const char * name ) [inline]
```

Set the name of a kernel object.

Parameters

<i>cap</i>	Capability which refers to the kernel object.
<i>name</i>	Name of the kernel object that is e.g. displayed in the kernel debugger.

This is a debugging facility, the call might be invalid.

Examples:

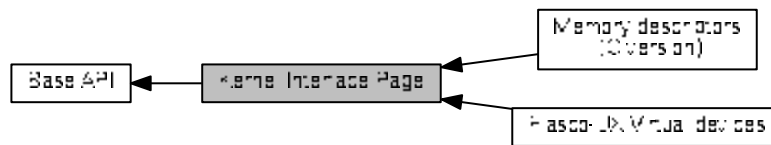
[examples/sys/aliens/main.c](#).

Definition at line [324](#) of file [debugger.h](#).

12.54 Kernel Interface Page

Kernel Interface Page.

Collaboration diagram for Kernel Interface Page:



Modules

- [Fiasco-UX Virtual devices](#)
Virtual hardware devices, provided by Fiasco-UX.
- [Memory descriptors \(C version\)](#)
C Interface for KIP memory descriptors.

Data Structures

- struct [l4_kernel_info_t](#)
L4 Kernel Interface Page.
- class [L4::Kip::Mem_desc](#)
Memory descriptors stored in the kernel interface page.

Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`
Kernel Info Page identifier ("L4μK").

Typedefs

- typedef struct [l4_kernel_info_t](#) [l4_kernel_info_t](#)
L4 Kernel Interface Page.
- typedef struct [l4_kernel_info_t](#) [l4_kernel_info_t](#)
L4 Kernel Interface Page.

Functions

- `l4_umword_t l4_kip_version (l4_kernel_info_t *kip) L4_NOTHROW`
Get the kernel version.
- `const char * l4_kip_version_string (l4_kernel_info_t *kip) L4_NOTHROW`
Get the kernel version string.
- `int l4_kernel_info_version_offset (l4_kernel_info_t *kip) L4_NOTHROW`
Return offset in bytes of version_strings relative to the KIP base.
- `l4_cpu_time_t l4_kip_clock (l4_kernel_info_t *kip) L4_NOTHROW`
Return clock value from the KIP.
- `l4_umword_t l4_kip_clock_lw (l4_kernel_info_t *kip) L4_NOTHROW`
Return least significant machine word of clock value from the KIP.

12.54.1 Detailed Description

Kernel Interface Page.

C interface for the Kernel Interface Page:

C++ interface for the Kernel Interface Page:

Include File

```
#include <l4/sys/kip>
```

Include File

```
#include <l4/sys/kip.h>
```

12.54.2 Function Documentation

12.54.2.1 l4_kernel_info_version_offset()

```
int l4_kernel_info_version_offset (
    l4_kernel_info_t * kip ) [inline]
```

Return offset in bytes of version_strings relative to the KIP base.

Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	--

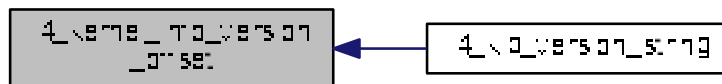
Returns

offset of version_strings relative to the KIP base address, in bytes.

Definition at line 134 of file [kip.h](#).

Referenced by [l4_kip_version_string\(\)](#).

Here is the caller graph for this function:



12.54.2.2 l4_kip_clock()

```
l4_cpu_time_t l4_kip_clock (
    l4_kernel_info_t * kip ) [inline]
```

Return clock value from the KIP.

Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	--

Returns

Value of the clock field in the KIP.

Definition at line 138 of file [kip.h](#).

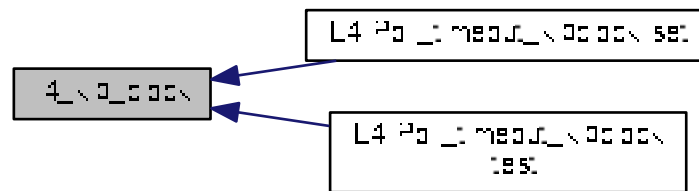
References [l4_mb\(\)](#).

Referenced by [L4::Poll_timeout_kipclock::set\(\)](#), and [L4::Poll_timeout_kipclock::test\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.54.2.3 l4_kip_clock_lw()

```
l4_umword_t l4_kip_clock_lw (
    l4_kernel_info_t * kip ) [inline]
```

Return least significant machine word of clock value from the KIP.

Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	--

Returns

Lower machine word of clock value from the KIP.

Definition at line 160 of file [kip.h](#).

References [l4_mb\(\)](#).

Here is the call graph for this function:



12.54.2.4 l4_kip_version()

```
l4_umword_t l4_kip_version (
    l4_kernel_info_t * kip ) [inline]
```

Get the kernel version.

Parameters

<i>kip</i>	Kernel Info Page.
------------	-------------------

Returns

Kernel version string. 0 if KIP could not be mapped.

Definition at line 126 of file [kip.h](#).

12.54.2.5 l4_kip_version_string()

```
const char * l4_kip_version_string (
    l4_kernel_info_t * kip ) [inline]
```

Get the kernel version string.

Parameters

<i>kip</i>	Kernel Info Page.
------------	-------------------

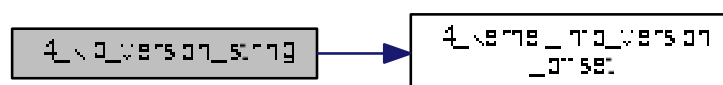
Returns

Kernel version string.

Definition at line 130 of file [kip.h](#).

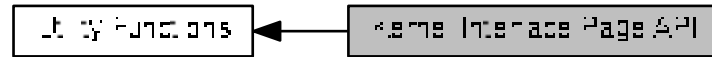
References [l4_kernel_info_version_offset\(\)](#).

Here is the call graph for this function:



12.55 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:



Files

- file [kip.h](#)

Macros

- `#define l4util_kip_for_each_feature(s)` for (s += strlen(s) + 1; *s; s += strlen(s) + 1)
Cycle through kernel features given in the KIP.

Functions

- `int l4util_kip_kernel_is_ux (l4_kernel_info_t *)`
Return whether the kernel is running native or under UX.
- `int l4util_kip_kernel_has_feature (l4_kernel_info_t *, const char *str)`
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t *)`
Return kernel ABI version.
- `l4_addr_t l4util_memdesc_vm_high (l4_kernel_info_t *kinfo)`
Return end of virtual memory.

12.55.1 Detailed Description

12.55.2 Macro Definition Documentation

12.55.2.1 l4util_kip_for_each_feature

```
#define l4util_kip_for_each_feature(
    s ) for ( s += strlen(s) + 1; *s; s += strlen(s) + 1)
```

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. `s` must be a character pointer (`char *`) initialized with `l4util_kip_version_string()`.

Definition at line 74 of file [kip.h](#).

12.55.3 Function Documentation

12.55.3.1 l4util_kip_kernel_abi_version()

```
unsigned long l4util_kip_kernel_abi_version (
    l4_kernel_info_t * )
```

Return kernel ABI version.

Returns

Kernel ABI version.

12.55.3.2 l4util_kip_kernel_has_feature()

```
int l4util_kip_kernel_has_feature (
    l4_kernel_info_t * ,
    const char * str )
```

Check if kernel supports a feature.

Parameters

<i>str</i>	Feature name to check.
------------	------------------------

Returns

1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

12.55.3.3 l4util_kip_kernel_is_ux()

```
int l4util_kip_kernel_is_ux (
    l4_kernel_info_t * )
```

Return whether the kernel is running native or under UX.

Returns whether the kernel is running natively or under UX. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

Returns

1 when running under UX, 0 if not running under UX

Examples:

[examples/sys/ux-vhw/main.c](#).

12.55.3.4 l4util_memdesc_vm_high()

```
l4_addr_t l4util_memdesc_vm_high (
    l4_kernel_info_t * kinfo )
```

Return end of virtual memory.

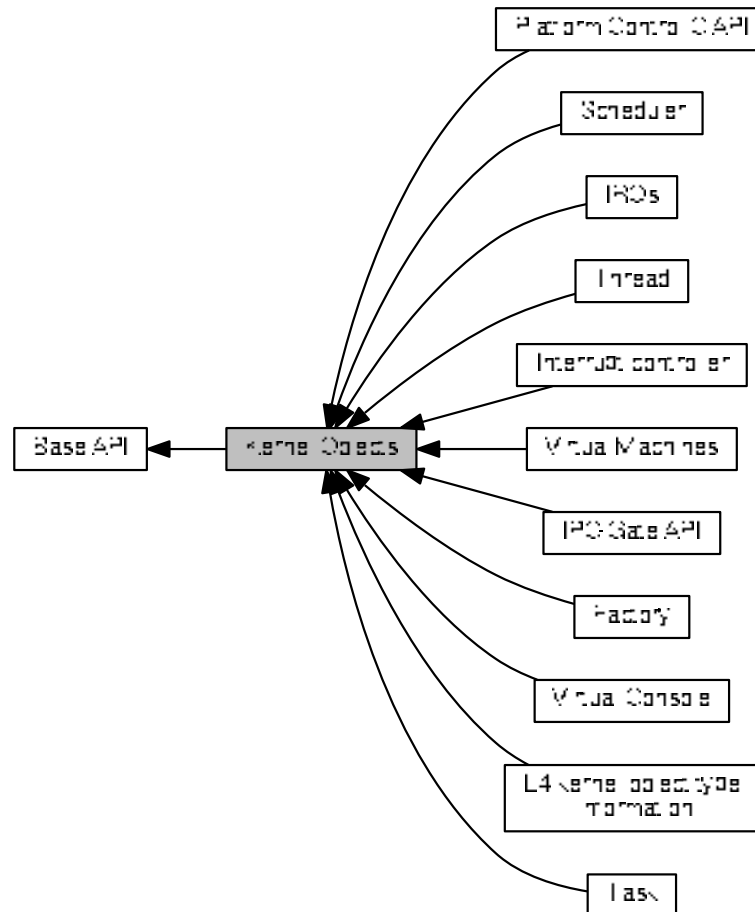
Returns

0 if memory descriptor could not be found, last address of address space otherwise

12.56 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:



Modules

- [Factory](#)
C factory interface to create kernel objects.
- [IPC-Gate API](#)
Secure communication object.
- [IRQs](#)
C IRQ interface.
- [Interrupt controller](#)
The C Icu interface.
- [L4 kernel object type information](#)
Type information for [L4](#) server objects that can be called via IPC.

- [Platform Control C API](#)
C interface for controlling platform-wide properties.
- [Scheduler](#)
C interface of the Scheduler kernel object.
- [Task](#)
C interface of the Task kernel object.
- [Thread](#)
Thread object.
- [Virtual Console](#)
Virtual console for simple character based input and output.
- [Virtual Machines](#)
Virtual Machine API.

Data Structures

- class [L4::Kobject](#)
Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
- class [L4::Vm](#)
Virtual machine.

12.56.1 Detailed Description

API of kernel objects.

Include File

```
#include <l4/sys/kernel_object.h>
```

12.57 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:



Kumem allocator utility C interface.

12.58 Kumem utilities

Collaboration diagram for Kumem utilities:



Functions

- `int L4Re::Util::kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) throw ()`
Allocate state area.

12.58.1 Detailed Description

12.58.2 Function Documentation

12.58.2.1 kumem_alloc()

```

int L4Re::Util::kumem_alloc (
    l4_addr_t * mem,
    unsigned pages_order,
    L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
    L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() ) throw ()

```

Allocate state area.

Parameters

out	<i>mem</i>	Pointer to memory that has been allocated.
	<i>pages_order</i>	Size to allocate, in log2 pages.
	<i>task</i>	Task to use for allocation.
	<i>rm</i>	Region manager to use for allocation.

Return values

0	for success
<0	error code on failure

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

12.59 L4 IPC Opcodes

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

Enumerations

- enum [L4_icu_opcode](#) {
[L4_ICU_OP_BIND](#), [L4_ICU_OP_UNBIND](#), [L4_ICU_OP_INFO](#), [L4_ICU_OP_MSI_INFO](#),
[L4_ICU_OP_UNMASK](#), [L4_ICU_OP_MASK](#), [L4_ICU_OP_SET_MODE](#) }
Opcodes to the ICU interface.
- enum [L4_ipc_gate_ops](#) { [L4_IPC_GATE_BIND_OP](#) = 0x10, [L4_IPC_GATE_GET_INFO_OP](#) = 0x11 }
Operations on the IPC-gate.
- enum [L4_platform_ctl_ops](#) { [L4_PLATFORM_CTL_SYS_SUSPEND_OP](#) = 0UL, [L4_PLATFORM_CTL_SYS_SHUTDOWN_OP](#) = 1UL, [L4_PLATFORM_CTL_CPU_ENABLE_OP](#) = 3UL, [L4_PLATFORM_CTL_CPU_DISABLE_OP](#) = 4UL }
Operations on platform-control objects.
- enum [L4_task_ops](#) {
[L4_TASK_MAP_OP](#) = 0UL, [L4_TASK_UNMAP_OP](#) = 1UL, [L4_TASK_CAP_INFO_OP](#) = 2UL, [L4_TASK_ADD_KU_MEM_OP](#) = 3UL,
[L4_TASK_LDT_SET_X86_OP](#) = 0x11UL }
Operations on task objects.
- enum [L4_thread_ops](#) {
[L4_THREAD_CONTROL_OP](#) = 0UL, [L4_THREAD_EX_REGS_OP](#) = 1UL, [L4_THREAD_SWITCH_OP](#) = 2UL, [L4_THREAD_STATS_OP](#) = 3UL,
[L4_THREAD_VCPU_RESUME_OP](#) = 4UL, [L4_THREAD_REGISTER_DELETE_IRQ_OP](#) = 5UL, [L4_THREAD_MODIFY_SENDER_OP](#) = 6UL, [L4_THREAD_VCPU_CONTROL_OP](#) = 7UL ,
[L4_THREAD_X86_GDT_OP](#) = 0x10UL, [L4_THREAD_ARM_TPIDRURO_OP](#) = 0x10UL, [L4_THREAD_AMD64_SET_SEGMENT_BASE_OP](#) = 0x12UL, [L4_THREAD_AMD64_GET_SEGMENT_INFO_OP](#) = 0x13UL,
[L4_THREAD_OPCODE_MASK](#) = 0xffff }
Operations on thread objects.
- enum [L4_vcon_ops](#) { [L4_VCON_WRITE_OP](#) = 0UL, [L4_VCON_READ_OP](#) = 1UL, [L4_VCON_SET_ATTR_OP](#) = 2UL, [L4_VCON_GET_ATTR_OP](#) = 3UL }
Operations on vcon objects.

12.59.1 Detailed Description

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

12.59.2 Enumeration Type Documentation

12.59.2.1 L4_icu_opcode

```
enum L4\_icu\_opcode
```

Opcodes to the ICU interface.

Enumerator

L4_ICU_OP_BIND	Bind opcode. See also l4_icu_bind()
L4_ICU_OP_UNBIND	Unbind opcode. See also l4_icu_unbind()
L4_ICU_OP_INFO	Info opcode. See also l4_icu_info()
L4_ICU_OP_MSI_INFO	Msi-info opcode. See also l4_icu_msi_info()
L4_ICU_OP_UNMASK	Unmask opcode. See also l4_icu_unmask()
L4_ICU_OP_MASK	Mask opcode. See also l4_icu_mask()
L4_ICU_OP_SET_MODE	Set-mode opcode. See also l4_icu_set_mode()

Definition at line 93 of file [icu.h](#).

12.59.2.2 L4_ipc_gate_ops

enum [L4_ipc_gate_ops](#)

Operations on the IPC-gate.

Enumerator

L4_IPC_GATE_BIND_OP	Bind operation.
L4_IPC_GATE_GET_INFO_OP	Info operation.

Definition at line 94 of file [ipc_gate.h](#).

12.59.2.3 L4_platform_ctl_ops

enum [L4_platform_ctl_ops](#)

Operations on platform-control objects.

See [L4_PROTO_PLATFORM_CTL](#) for the protocol type to use for messages to platform-control objects.

Enumerator

L4_PLATFORM_CTL_SYS_SUSPEND_OP	Suspend.
L4_PLATFORM_CTL_SYS_SHUTDOWN_OP	shutdown/reboot
L4_PLATFORM_CTL_CPU_ENABLE_OP	enable an offline CPU
L4_PLATFORM_CTL_CPU_DISABLE_OP	disable an online CPU

Definition at line [135](#) of file [platform_control.h](#).

12.59.2.4 L4_task_ops

enum [L4_task_ops](#)

Operations on task objects.

Enumerator

L4_TASK_MAP_OP	Map.
L4_TASK_UNMAP_OP	Unmap.
L4_TASK_CAP_INFO_OP	Cap info.
L4_TASK_ADD_KU_MEM_OP	Add kernel-user memory.
L4_TASK_LDT_SET_X86_OP	x86: LDT set

Definition at line [276](#) of file [task.h](#).

12.59.2.5 L4_thread_ops

enum [L4_thread_ops](#)

Operations on thread objects.

Enumerator

L4_THREAD_CONTROL_OP	Control operation.
----------------------	--------------------

Enumerator

L4_THREAD_EX_REGS_OP	Exchange registers operation.
L4_THREAD_SWITCH_OP	Do a thread switch.
L4_THREAD_STATS_OP	Thread statistics.
L4_THREAD_VCPU_RESUME_OP	VCPU resume.
L4_THREAD_REGISTER_DELETE_IRQ_OP	Register an IPC-gate deletion IRQ.
L4_THREAD_MODIFY_SENDER_OP	Modify all senders IDs that match the given pattern.
L4_THREAD_VCPU_CONTROL_OP	Enable / disable VCPU feature.
L4_THREAD_X86_GDT_OP	Gdt.
L4_THREAD_ARM_TPIDRURO_OP	Set TPIDRURO register.
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP	Set segment base.
L4_THREAD_AMD64_GET_SEGMENT_INFO_OP	Get segment information.
L4_THREAD_OPCODE_MASK	Mask for opcodes.

Definition at line 612 of file [thread.h](#).

12.59.2.6 L4_vcon_ops

enum [L4_vcon_ops](#)

Operations on vcon objects.

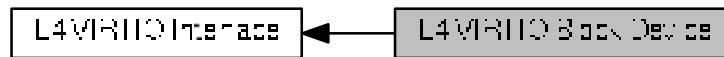
Enumerator

L4_VCON_WRITE_OP	Write.
L4_VCON_READ_OP	Read.
L4_VCON_SET_ATTR_OP	Get console attributes.
L4_VCON_GET_ATTR_OP	Set console attributes.

Definition at line 264 of file [vcon.h](#).

12.60 L4 VIRTIO Block Device

Collaboration diagram for L4 VIRTIO Block Device:



Data Structures

- struct [l4virtio_block_header_t](#)
Header structure of a request for a block device.
- struct [l4virtio_block_discard_t](#)
Structure used for the write zeroes and discard commands.
- struct [l4virtio_block_config_t](#)
Device configuration for block devices.

Typedefs

- typedef struct [l4virtio_block_header_t](#) [l4virtio_block_header_t](#)
Header structure of a request for a block device.
- typedef struct [l4virtio_block_discard_t](#) [l4virtio_block_discard_t](#)
Structure used for the write zeroes and discard commands.
- typedef struct [l4virtio_block_config_t](#) [l4virtio_block_config_t](#)
Device configuration for block devices.

Enumerations

- enum [L4virtio_block_operations](#) {
[L4VIRTIO_BLOCK_T_IN](#) = 0, [L4VIRTIO_BLOCK_T_OUT](#) = 1, [L4VIRTIO_BLOCK_T_FLUSH](#) = 4, [L4VIRTIO_BLOCK_T_GET_ID](#) = 8,
[L4VIRTIO_BLOCK_T_DISCARD](#) = 11, [L4VIRTIO_BLOCK_T_WRITE_ZEROES](#) = 13 }
Kinds of operation over a block device.
- enum [L4virtio_block_status](#) { [L4VIRTIO_BLOCK_S_OK](#) = 0, [L4VIRTIO_BLOCK_S_IOERR](#) = 1, [L4VIRTIO_BLOCK_S_UNSUPP](#) = 2 }
Status of a finished block request.

12.60.1 Detailed Description

12.60.2 Enumeration Type Documentation

12.60.2.1 L4virtio_block_operations

enum [L4virtio_block_operations](#)

Kinds of operation over a block device.

Enumerator

L4VIRTIO_BLOCK_T_IN	Read from device.
L4VIRTIO_BLOCK_T_OUT	Write to device.
L4VIRTIO_BLOCK_T_FLUSH	Flush data to disk.
L4VIRTIO_BLOCK_T_GET_ID	Get device ID.
L4VIRTIO_BLOCK_T_DISCARD	Discard a range of sectors.
L4VIRTIO_BLOCK_T_WRITE_ZEROES	Write zeroes to a range of sectors.

Definition at line 31 of file [virtio_block.h](#).

12.60.2.2 L4virtio_block_status

enum [L4virtio_block_status](#)

Status of a finished block request.

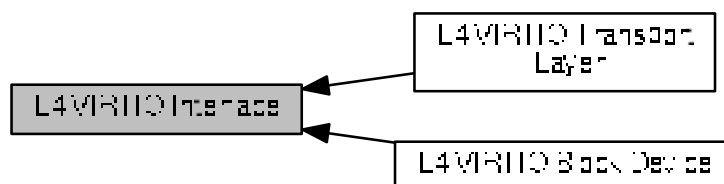
Enumerator

L4VIRTIO_BLOCK_S_OK	Request finished successfully.
L4VIRTIO_BLOCK_S_IOERR	IO error on device.
L4VIRTIO_BLOCK_S_UNSUPP	Operation is not supported.

Definition at line 44 of file [virtio_block.h](#).

12.61 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:



Modules

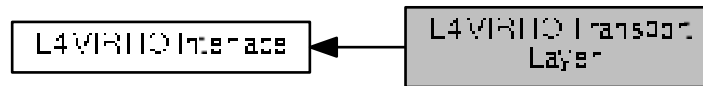
- [L4 VIRTIO Block Device](#)
- [L4 VIRTIO Transport Layer](#)
L4 specific VIRTIO Transport layer.

12.61.1 Detailed Description

12.62 L4 VIRTIO Transport Layer

L4 specific VIRTIO Transport layer.

Collaboration diagram for L4 VIRTIO Transport Layer:



Namespaces

- [L4virtio](#)
L4-VIRTIO Transport C++ API.

Data Structures

- struct [l4virtio_config_hdr_t](#)
L4-VIRTIO config header, provided in shared data space.
- struct [l4virtio_config_queue_t](#)
Queue configuration entry.

Typedefs

- typedef struct [l4virtio_config_hdr_t](#) [l4virtio_config_hdr_t](#)
L4-VIRTIO config header, provided in shared data space.
- typedef struct [l4virtio_config_queue_t](#) [l4virtio_config_queue_t](#)
Queue configuration entry.

Enumerations

- enum [L4_virtio_protocol](#)
L4-VIRTIO protocol number.
- enum [L4_virtio_opcodes](#) { [L4VIRTIO_OP_SET_STATUS](#) = 0, [L4VIRTIO_OP_CONFIG_QUEUE](#), [L4VIRTIO_OP_REGISTER_IFACE](#), [L4VIRTIO_OP_REGISTER_DS](#) }
L4-VIRTIO opcodes.
- enum [L4virtio_device_ids](#) {
[L4VIRTIO_ID_NET](#) = 1, [L4VIRTIO_ID_BLOCK](#) = 2, [L4VIRTIO_ID_CONSOLE](#) = 3, [L4VIRTIO_ID_RNG](#) = 4,
[L4VIRTIO_ID_BALLOON](#) = 5, [L4VIRTIO_ID_RPMSG](#) = 7, [L4VIRTIO_ID_SCSI](#) = 8, [L4VIRTIO_ID_9P](#) = 9,
[L4VIRTIO_ID_RPROC_SERIAL](#) = 11, [L4VIRTIO_ID_CAIF](#) = 12, [L4VIRTIO_ID_GPU](#) = 16, [L4VIRTIO_ID_INPUT](#) = 18,
[L4VIRTIO_ID_VSOCK](#) = 19, [L4VIRTIO_ID_CRYPT](#) = 20, [L4VIRTIO_ID_SOCK](#) = 0x9999 }
Virtio device IDs as reported in the driver's config space.

- enum `L4virtio_device_status` {
`L4VIRTIO_STATUS_ACKNOWLEDGE` = 1, `L4VIRTIO_STATUS_DRIVER` = 2, `L4VIRTIO_STATUS_DRIVER_OK` = 4, `L4VIRTIO_STATUS_FEATURES_OK` = 8,
`L4VIRTIO_STATUS_FAILED` = 0x80 }
Virtio device status bits.
- enum `L4virtio_feature_bits` { `L4VIRTIO_FEATURE_VERSION_1` = 32, `L4VIRTIO_FEATURE_CMD_CONFIG` = 224 }
L4virtio-specific feature bits.
- enum `L4_virtio_irq_status` { `L4VIRTIO_IRQ_STATUS_VRING` = 1, `L4VIRTIO_IRQ_STATUS_CONFIG` = 2 }
VIRTIO IRQ status codes (l4virtio_config_hdr_t::irq_status).
- enum `L4_virtio_cmd` { `L4VIRTIO_CMD_NONE` = 0x00000000, `L4VIRTIO_CMD_SET_STATUS` = 0x01000000, `L4VIRTIO_CMD_CFG_QUEUE` = 0x02000000, `L4VIRTIO_CMD_MASK` = 0xff000000 }
Virtio commands for device configuration.

Functions

- `l4virtio_config_queue_t * l4virtio_config_queues (l4virtio_config_hdr_t const *cfg)`
Get the pointer to the first queue config.
- `void * l4virtio_device_config (l4virtio_config_hdr_t const *cfg)`
Get the pointer to the device configuration.
- `void l4virtio_set_feature (l4_uint32_t *feature_map, unsigned feat)`
Set the given feature bit in a feature map.
- `void l4virtio_clear_feature (l4_uint32_t *feature_map, unsigned feat)`
Clear the given feature bit in a feature map.
- `unsigned l4virtio_get_feature (l4_uint32_t *feature_map, unsigned feat)`
Check if the given bit in a feature map is set.
- `int l4virtio_set_status (l4_cap_idx_t cap, unsigned status) L4_NOTHROW`
Write the VIRTIO status register.
- `int l4virtio_config_queue (l4_cap_idx_t cap, unsigned queue) L4_NOTHROW`
Trigger queue configuration of the given queue.
- `int l4virtio_register_ds (l4_cap_idx_t cap, l4_cap_idx_t ds_cap, l4_uint64_t base, l4_umword_t offset, l4_umword_t size) L4_NOTHROW`
Register a shared data space with VIRTIO host.
- `int l4virtio_register_iface (l4_cap_idx_t cap, l4_cap_idx_t guest_irq, l4_cap_idx_t host_irq, l4_cap_idx_t config_ds) L4_NOTHROW`
Register client to the L4-VIRTIO device.
- `int l4virtio_device_config_ds (l4_cap_idx_t cap, l4_cap_idx_t config_ds, l4_addr_t *ds_offset) L4_NOTHROW`
Get the dataspace with the L4virtio configuration page.
- `int l4virtio_device_notification_irq (l4_cap_idx_t cap, unsigned index, l4_cap_idx_t irq) L4_NOTHROW`
Get the notification interrupt corresponding to the given index.

12.62.1 Detailed Description

L4 specific VIRTIO Transport layer.

The L4 specific VIRTIO Transport layer is based on `L4Re::Dataspace` as shared memory and `L4::Irq` for signaling. The VIRTIO configuration space is mostly based on a shared memory implementation too and accompanied by two IPC functions to synchronize the configuration between device and driver.

12.62.2 Typedef Documentation

12.62.2.1 l4virtio_config_queue_t

```
typedef struct l4virtio_config_queue_t l4virtio_config_queue_t
```

Queue configuration entry.

An array of such entries is available at the [l4virtio_config_hdr_t::queues_offset](#) in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by `L4VIRTIO_CMD_CFG_QUEUE`.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful `L4VIRTIO_CMD_CFG_QUEUE` or after a `config_queue` IPC. Note, the change might also have immediate effect, depending on the device implementation.

12.62.3 Enumeration Type Documentation

12.62.3.1 L4_virtio_cmd

```
enum L4_virtio_cmd
```

Virtio commands for device configuration.

Enumerator

<code>L4VIRTIO_CMD_NONE</code>	No command pending.
<code>L4VIRTIO_CMD_SET_STATUS</code>	Set the status register.
<code>L4VIRTIO_CMD_CFG_QUEUE</code>	Configure a queue.
<code>L4VIRTIO_CMD_MASK</code>	Mask to get command bits.

Definition at line 116 of file [virtio.h](#).

12.62.3.2 L4_virtio_irq_status

```
enum L4_virtio_irq_status
```

VIRTIO IRQ status codes (`l4virtio_config_hdr_t::irq_status`).

Note

`l4virtio_config_hdr_t::irq_status` is currently unused.

Enumerator

<code>L4VIRTIO_IRQ_STATUS_VRING</code>	VRING IRQ pending flag.
<code>L4VIRTIO_IRQ_STATUS_CONFIG</code>	CONFIG IRQ pending flag.

Definition at line 107 of file [virtio.h](#).

12.62.3.3 L4_virtio_opcodes

```
enum L4_virtio_opcodes
```

L4-VIRTIO opcodes.

Enumerator

<code>L4VIRTIO_OP_SET_STATUS</code>	Set status register in device config.
<code>L4VIRTIO_OP_CONFIG_QUEUE</code>	Set queue config in device config.
<code>L4VIRTIO_OP_REGISTER_IFACE</code>	Register a transport driver to the device.
<code>L4VIRTIO_OP_REGISTER_DS</code>	Register a data space as transport memory.

Definition at line 55 of file [virtio.h](#).

12.62.3.4 L4virtio_device_ids

```
enum L4virtio_device_ids
```

Virtio device IDs as reported in the driver's config space.

Enumerator

<code>L4VIRTIO_ID_NET</code>	Virtual ethernet card.
<code>L4VIRTIO_ID_BLOCK</code>	General block device.
<code>L4VIRTIO_ID_CONSOLE</code>	Simple device for data IO via ports.
<code>L4VIRTIO_ID_RNG</code>	Entropy source.
<code>L4VIRTIO_ID_BALLOON</code>	Memory ballooning device.
<code>L4VIRTIO_ID_RPMMSG</code>	Device using rpmmsg protocol.
<code>L4VIRTIO_ID_SCSI</code>	SCSI host device.
<code>L4VIRTIO_ID_9P</code>	Device using 9P transport protocol.

Enumerator

L4VIRTIO_ID_RPROC_SERIAL	Rproc serial device.
L4VIRTIO_ID_CAIF	Device using CAIF network protocol.
L4VIRTIO_ID_GPU	GPU.
L4VIRTIO_ID_INPUT	Input.
L4VIRTIO_ID_VSOCK	Vsock transport.
L4VIRTIO_ID_CRYPT	Crypto.
L4VIRTIO_ID_SOCKET	Inofficial socket device.

Definition at line 64 of file [virtio.h](#).

12.62.3.5 L4virtio_device_status

enum [L4virtio_device_status](#)

Virtio device status bits.

Enumerator

L4VIRTIO_STATUS_ACKNOWLEDGE	Guest OS has found device.
L4VIRTIO_STATUS_DRIVER	Guest OS knows how to drive device.
L4VIRTIO_STATUS_DRIVER_OK	Driver is set up.
L4VIRTIO_STATUS_FEATURES_OK	Driver has acknowledged feature set.
L4VIRTIO_STATUS_FAILED	Fatal error in driver or device.

Definition at line 85 of file [virtio.h](#).

12.62.3.6 L4virtio_feature_bits

enum [L4virtio_feature_bits](#)

L4virtio-specific feature bits.

Enumerator

L4VIRTIO_FEATURE_VERSION_1	Virtio protocol version 1 supported. Must be 1 for L4virtio .
L4VIRTIO_FEATURE_CMD_CONFIG	Status and queue config are set via cmd field instead of via IPC.

Definition at line 95 of file [virtio.h](#).

12.62.4 Function Documentation

12.62.4.1 l4virtio_config_queue()

```
int l4virtio_config_queue (
    l4_cap_idx_t cap,
    unsigned queue )
```

Trigger queue configuration of the given queue.

Parameters

<i>cap</i>	Capability to the VIRTIO host.
------------	--------------------------------

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

Parameters

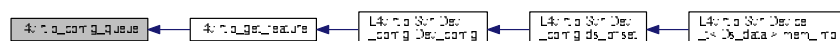
<i>queue</i>	Queue index for the queue to be configured.
--------------	---

Return values

0	on success.
-L4_EIO	The queue's status is invalid.
-L4_ERANGE	The queue index exceeds the number of queues.
-L4_EINVAL	Otherwise.

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.62.4.2 l4virtio_config_queues()

```
l4virtio_config_queue_t* l4virtio_config_queues (
    l4virtio_config_hdr_t const * cfg ) [inline]
```

Get the pointer to the first queue config.

Parameters

<i>cfg</i>	Pointer to the config header.
------------	-------------------------------

Returns

pointer to queue config of queue 0.

Definition at line 237 of file [virtio.h](#).

12.62.4.3 l4virtio_device_config()

```
void* l4virtio_device_config (
    l4virtio_config_hdr_t const * cfg ) [inline]
```

Get the pointer to the device configuration.

Parameters

<i>cfg</i>	Pointer to the config header.
------------	-------------------------------

Returns

pointer to device configuration structure.

Definition at line 248 of file [virtio.h](#).

12.62.4.4 l4virtio_device_config_ds()

```
int l4virtio_device_config_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t config_ds,
    l4_addr_t * ds_offset )
```

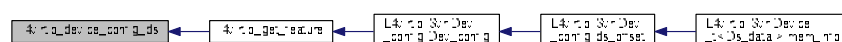
Get the dataspace with the [L4virtio](#) configuration page.

Parameters

<i>cap</i>	Capability to the L4-VIRTIO host
<i>config_ds</i>	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.
<i>ds_offset</i>	Offset into the dataspace where the device configuration structure starts.

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.62.4.5 l4virtio_device_notification_irq()

```
int l4virtio_device_notification_irq (
    l4_cap_idx_t cap,
    unsigned index,
    l4_cap_idx_t irq )
```

Get the notification interrupt corresponding to the given index.

Parameters

	<i>cap</i>	Capability to the L4-VIRTIO host
	<i>index</i>	Index of the interrupt.
out	<i>irq</i>	Triggerable for the given index.

Return values

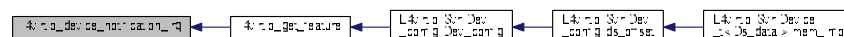
<i>L4_EOK</i>	Success.
<i>L4_ENOSYS</i>	IRQ notification not supported by device.
<i><0</i>	Other error.

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.62.4.6 l4virtio_register_ds()

```
int l4virtio_register_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t ds_cap,
    l4_uint64_t base,
    l4_umword_t offset,
    l4_umword_t size )
```

Register a shared data space with VIRTIO host.

Parameters

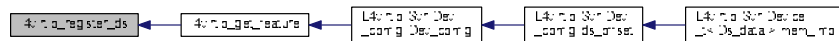
<i>cap</i>	Capability to the VIRTIO host
<i>ds_cap</i>	Data-space capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
<i>base</i>	VIRTIO guest physical start address of shared memory region
<i>offset</i>	Offset within the data space that is attached to the given <i>base</i> in the guest physical memory.
<i>size</i>	Size of the memory region in the guest

Returns

0 on success, < 0 on error

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.62.4.7 l4virtio_register_iface()

```

int l4virtio_register_iface (
    l4_cap_idx_t cap,
    l4_cap_idx_t guest_irq,
    l4_cap_idx_t host_irq,
    l4_cap_idx_t config_ds )

```

Register client to the L4-VIRTIO device.

Parameters

<i>cap</i>	Capability to the L4-VIRTIO host
<i>guest_irq</i>	IRQ capability for valid IRQ object for device-to-driver notifications.
<i>host_irq</i>	Capability selector for receiving the driver-to-device notifications IRQ capability.
<i>config_ds</i>	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.

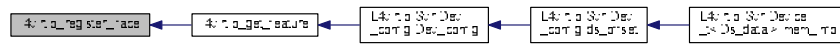
Return values

<i>L4_EOK</i>	Success.
<i>-L4_ENOSYS</i>	This interface is no longer supported by the server. Use <code>get_device_config()</code> etc. instead.

Deprecated Use `device_config()`, `device_notification_irq()` and `lcu::bind()` instead.

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.62.4.8 l4virtio_set_status()

```
int l4virtio_set_status (
    l4_cap_idx_t cap,
    unsigned status )
```

Write the VIRTIO status register.

Parameters

<i>cap</i>	Capability to the VIRTIO host
<i>status</i>	Status word to write to the VIRTIO status.

Returns

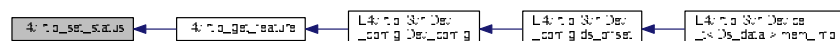
0 on success, <0 on error.

Note

All other registers are accessed via shared memory.

Referenced by [l4virtio_get_feature\(\)](#).

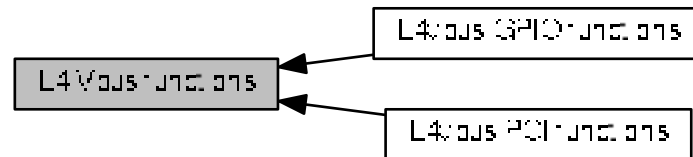
Here is the caller graph for this function:



12.63 L4 Vbus functions

C interface of the Vbus API.

Collaboration diagram for L4 Vbus functions:



Modules

- [L4vbus GPIO functions](#)
- [L4vbus PCI functions](#)

Enumerations

- enum [L4vbus_dma_domain_assign_flags](#) { [L4VBUS_DMAD_UNBIND](#) = 0, [L4VBUS_DMAD_BIND](#) = 1, [L4VBUS_DMAD_L4RE_DMA_SPACE](#) = 0, [L4VBUS_DMAD_KERNEL_DMA_SPACE](#) = 2 }
- Flags for [l4vbus_assign_dma_domain\(\)](#).

Functions

- int [l4vbus_get_device_by_hid](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) parent, [l4vbus_device_handle_t](#) *child, char const *hid, int depth, [l4vbus_device_t](#) *devinfo)
Find a device by the human interface identifier (HID).
- int [l4vbus_get_next_device](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) parent, [l4vbus_device_handle_t](#) *child, int depth, [l4vbus_device_t](#) *devinfo)
Find next child following *child*.
- int [l4vbus_get_device](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) dev, [l4vbus_device_t](#) *devinfo)
Obtain detailed information about a Vbus device.
- int [l4vbus_get_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) dev, int res_idx, [l4vbus_resource_t](#) *res)
Obtain the resource description of an individual device resource.
- int [l4vbus_is_compatible](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) dev, char const *cid)
Check if the given device has a compatibility ID (CID) or HID that matches *cid*.
- int [l4vbus_get_hid](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) dev, char *hid, unsigned long max_len)
Get the HID (hardware identifier) of a device.
- int [l4vbus_request_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res, int flags)
Request a resource of a specific type.
- int [l4vbus_assign_dma_domain](#) ([l4_cap_idx_t](#) vbus, unsigned domain_id, unsigned flags, [l4_cap_idx_t](#) dma_space)
Bind or unbind a kernel DMA space ([L4::Task](#)) or a [L4Re::Dma_space](#) to a DMA domain.
- int [l4vbus_release_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res)
Release a previously requested resource.
- int [l4vbus_vicu_get_cap](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) icu, [l4_cap_idx_t](#) cap)
Get capability of ICU.

12.63.1 Detailed Description

C interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Include File

```
#include <l4/vbus/vbus.h>
```

Refer to L4vbus for the C++ API.

12.63.2 Enumeration Type Documentation

12.63.2.1 L4vbus_dma_domain_assign_flags

```
enum L4vbus_dma_domain_assign_flags
```

Flags for [l4vbus_assign_dma_domain\(\)](#).

Enumerator

L4VBUS_DMAD_UNBIND	Unbind the given DMA space from the DMA domain.
L4VBUS_DMAD_BIND	Bind the given DMA space to the DMA domain.
L4VBUS_DMAD_L4RE_DMA_SPACE	The given DMA space is an L4Re::Dma_space .
L4VBUS_DMAD_KERNEL_DMA_SPACE	The given DMA space is a kernel DMA space (L4::Task)

Definition at line 153 of file [vbus.h](#).

12.63.3 Function Documentation

12.63.3.1 l4vbus_assign_dma_domain()

```
int l4vbus_assign_dma_domain (
    l4_cap_idx_t vbus,
    unsigned domain_id,
    unsigned flags,
    l4_cap_idx_t dma_space )
```

Bind or unbind a kernel DMA space ([L4::Task](#)) or a [L4Re::Dma_space](#) to a DMA domain.

Parameters

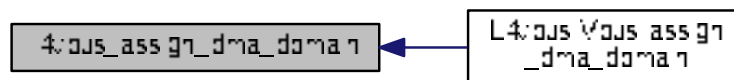
<i>vbus</i>	Capability of the system bus
<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of L4vbus_dma_domain_assign_flags .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must either be an L4Re::Dma_space or a kernel DMA space (L4::Task created with <code>L4_PROTO_DMA_SPACE</code>) and the type must be reflected in the <i>flags</i> .

Return values

<i>0</i>	Operation completed successfully.
<i>-L4_ENOENT</i>	The vbus does not support a global DMA domain or no DMA domain could be found.
<i>-L4_EINVAL</i>	Invalid argument used.
<i>-L4_EBUSY</i>	DMA domain is already active, this means another DMA space is already assigned.

Referenced by [L4vbus::Vbus::assign_dma_domain\(\)](#).

Here is the caller graph for this function:



12.63.3.2 l4vbus_get_device()

```

int l4vbus_get_device (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    l4vbus_device_t * devinfo )
  
```

Obtain detailed information about a Vbus device.

Parameters

	<i>vbus</i>	Capability of the vbus to which the device is connected.
	<i>dev</i>	Device handle of the device from which to retrieve the details.
out	<i>devinfo</i>	Information structure which contains details about the device. The pointer might be NULL after a successful call.

Return values

0	Success.
-L4_ENODEV	No device with the given device handle <code>dev</code> could be found.

Referenced by [L4vbus::Device::device\(\)](#).

Here is the caller graph for this function:



12.63.3.3 l4vbus_get_device_by_hid()

```

int l4vbus_get_device_by_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t parent,
    l4vbus_device_handle_t * child,
    char const * hid,
    int depth,
    l4vbus_device_t * devinfo )

```

Find a device by the human interface identifier (HID).

Parameters

<i>vbus</i>	Capability of the system bus
<i>parent</i>	Handle to the parent to start the search This function searches the <i>vbus</i> for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a *vbus*. In order to find all matching devices this function has to be called repeatedly with *child* pointing to the device found in the previous iteration. The iteration starts at *child* that might be any device node in the tree.

Parameters

<i>in, out</i>	<i>child</i>	Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default (L4VBUS_NULL). If a matching device is found its handle is returned through this parameter.
	<i>hid</i>	HID of the device
	<i>depth</i>	Maximum depth for the recursive lookup

Parameters

out	<i>devinfo</i>	Device information structure (might be NULL)
-----	----------------	--

Return values

<code>>=</code>	0 A device with the given HID was found.
<code>-L4_ENOENT</code>	No device with the given HID could be found on the vbus.
<code>-L4_EINVAL</code>	Invalid or no HID provided.
<code>-L4_ENODEV</code>	Function called on a non-existing device.

12.63.3.4 l4vbus_get_hid()

```
int l4vbus_get_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char * hid,
    unsigned long max_len )
```

Get the HID (hardware identifier) of a device.

Parameters

<i>vbus</i>	Capability of the system bus
<i>dev</i>	Handle of the device
<i>hid</i>	Pointer to a buffer for the HID string
<i>max_len</i>	The size of the buffer (<i>hid</i>)

Returns

the length of the HID string on success, else failure

12.63.3.5 l4vbus_get_next_device()

```
int l4vbus_get_next_device (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t parent,
    l4vbus_device_handle_t * child,
    int depth,
    l4vbus_device_t * devinfo )
```

Find next child following *child*.

Parameters

	<i>vbus</i>	Capability of the system bus
	<i>parent</i>	Handle to the parent device (use 0 for the system bus)
	<i>child</i>	Handle to the child device (use 0 to get the first child)
	<i>depth</i>	Depth to look for
out	<i>devinfo</i>	device information (might be NULL)

Returns

0 on success, else failure

12.63.3.6 l4vbus_get_resource()

```
int l4vbus_get_resource (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    int res_idx,
    l4vbus_resource_t * res )
```

Obtain the resource description of an individual device resource.

Parameters

	<i>vbus</i>	Capability of the vbus to which the device is connected.
	<i>dev</i>	Device handle of the device on the vbus. The device handle can be obtained by using the l4vbus_get_device_by_hid() and l4vbus_get_next_device() functions.
	<i>res_idx</i>	Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the l4vbus_device_t structure that is returned by L4vbus::Device::device_by_hid() and L4vbus::Device::next_device() .
out	<i>res</i>	Descriptor of the resource.

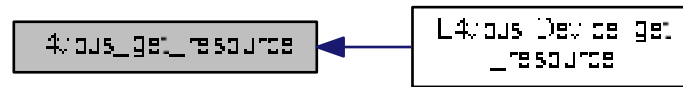
This function returns the resource descriptor of an individual device resource selected by the *res_idx* parameter.

Return values

0	Success.
-L4_ENOENT	Invalid resource index <i>res_idx</i> .

Referenced by [L4vbus::Device::get_resource\(\)](#).

Here is the caller graph for this function:



12.63.3.7 l4vbus_is_compatible()

```

int l4vbus_is_compatible (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char const * cid )
  
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

Parameters

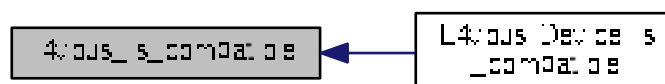
<i>vbus</i>	Capability of the system bus
<i>dev</i>	device handle for which the CID shall be tested
<i>cid</i>	the compatibility ID to test

Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Referenced by [L4vbus::Device::is_compatible\(\)](#).

Here is the caller graph for this function:



12.63.3.8 l4vbus_release_resource()

```
int l4vbus_release_resource (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res )
```

Release a previously requested resource.

Parameters

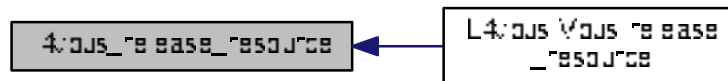
<i>vbus</i>	Capability of the system bus.
<i>res</i>	Descriptor of the resource.

Returns

0 on success, else failure

Referenced by [L4vbus::Vbus::release_resource\(\)](#).

Here is the caller graph for this function:



12.63.3.9 l4vbus_request_resource()

```
int l4vbus_request_resource (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res,
    int flags )
```

Request a resource of a specific type.

Parameters

<i>vbus</i>	Capability of the system bus
<i>res</i>	Descriptor of the resource
<i>flags</i>	Optional flags

Returns

0 on success, else failure

If any resource is found that contains the requested type and addresses this resource is returned.

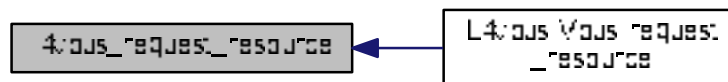
Flags are only relevant to control the memory caching. If io-memory is requested.

Returns

0 on success, else failure

Referenced by [L4vbus::Vbus::request_resource\(\)](#).

Here is the caller graph for this function:

**12.63.3.10 l4vbus_vicu_get_cap()**

```
int l4vbus_vicu_get_cap (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t icu,
    l4_cap_idx_t cap )
```

Get capability of ICU.

Parameters

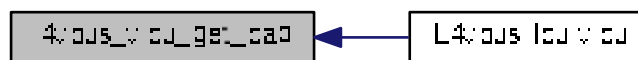
<i>vbus</i>	Capability of the system bus.
<i>icu</i>	ICU device handle.
<i>cap</i>	Capability slot for the capability.

Returns

0 on success, else failure

Referenced by [L4vbus::Icu::vicu\(\)](#).

Here is the caller graph for this function:



12.64 L4 kernel object type information

Type information for [L4](#) server objects that can be called via IPC.

Collaboration diagram for L4 kernel object type information:



Data Structures

- struct [L4::Type_info](#)
Dynamic Type Information for [L4Re](#) Interfaces.
- struct [L4::Kobject_typeid< T >](#)
Meta object for handling access to type information of Kobjects.
- class [L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >](#)
Helper class to create an [L4Re](#) interface class that is derived from a single base class.
- class [L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >](#)
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).
- struct [L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >](#)
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).
- struct [L4::Kobject_x< Derived, ARGS >](#)
Generic [Kobject](#) inheritance template.

Functions

- template<typename T >
[Type_info](#) const * [L4::kobject_typeid](#) ()
Get the [L4::Type_info](#) for the [L4Re](#) interface given in T.

12.64.1 Detailed Description

Type information for [L4](#) server objects that can be called via IPC.

This type information consists of inheritance information, the protocol number assigned to an interface as well as the demand on server-side resources.

12.64.2 Function Documentation

12.64.2.1 kobject_typeid()

```
template<typename T >
Type\_info const* L4::kobject\_typeid ( ) [inline]
```

Get the [L4::Type_info](#) for the [L4Re](#) interface given in T.

Template Parameters

<i>T</i>	The type (L4Re interface) for which the information shall be returned.
----------	---

Returns

A pointer to the [L4::Type_info](#) structure for *T*.

Definition at line 692 of file [__typeinfo.h](#).

References [L4::Kobject_typeid< T >::id\(\)](#), and [L4::PROTO_ANY](#).

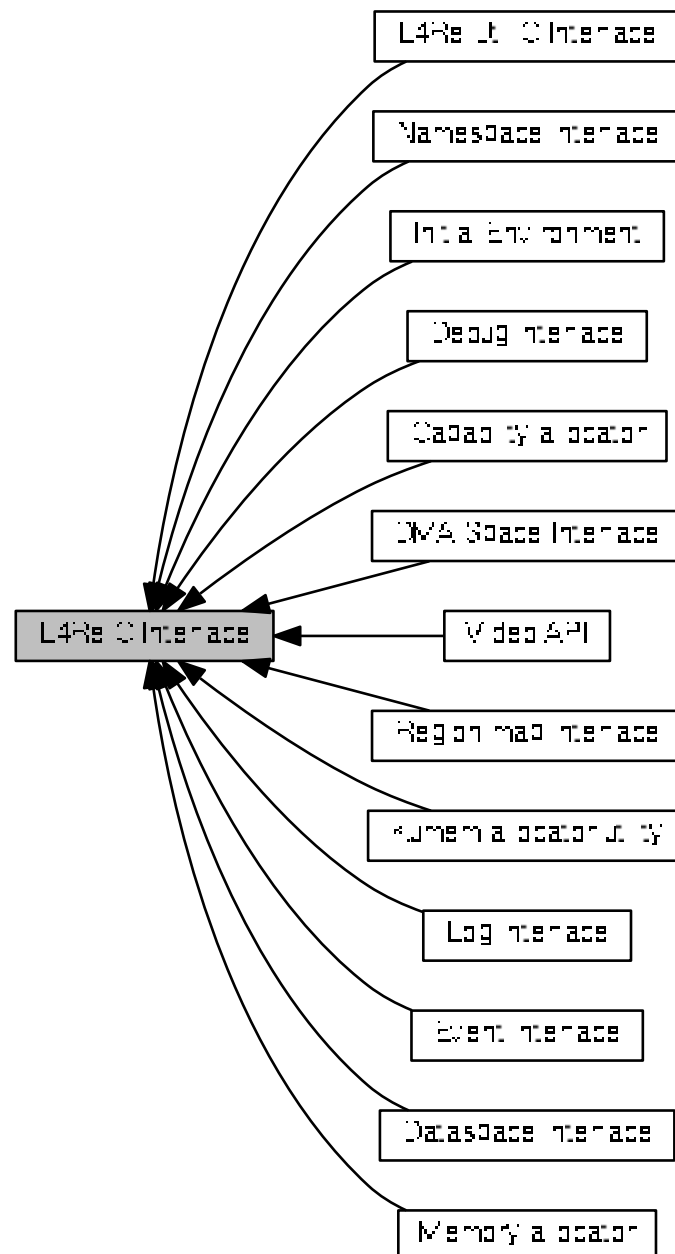
Here is the call graph for this function:



12.65 L4Re C Interface

Documentation for the [L4Re C Interface](#).

Collaboration diagram for L4Re C Interface:



Modules

- [Capability allocator](#)

- [Capability allocator C interface.](#)
- [DMA Space Interface](#)
DMA Space C interface.
- [Dataspace interface](#)
Dataspace C interface.
- [Debug interface](#)
- [Event interface](#)
Event C interface.
- [Initial Environment](#)
C interface of the initial environment that is provided to an [L4](#) task.
- [Kumem allocator utility](#)
Kumem allocator utility C interface.
- [L4Re Util C Interface](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C.
- [Log interface](#)
Log C interface.
- [Memory allocator](#)
Memory allocator C interface.
- [Namespace interface](#)
Namespace C interface.
- [Region map interface](#)
Region map C interface.
- [Video API](#)

Functions

- long [l4re_inhibitor_acquire](#) ([l4_cap_idx_t](#) cap, [l4_umword_t](#) id, char const *reason)
Inhibitor C interface.

12.65.1 Detailed Description

Documentation for the [L4Re](#) C Interface.

The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

12.65.2 Function Documentation

12.65.2.1 [l4re_inhibitor_acquire\(\)](#)

```
long l4re_inhibitor_acquire (
    l4\_cap\_idx\_t cap,
    l4\_umword\_t id,
    char const * reason )
```

Inhibitor C interface.

Acquire an inhibitor lock.

Parameters

<i>cap</i>	Capability for the Inhibitor object (
------------	---------------------------------------

See also

[L4Re::Inhibitor](#))

Parameters

<i>id</i>	ID of the inhibitor lock that shall be acquired.
<i>reason</i>	Reason why the inhibitor lock is acquired. (Used for informing the user or debugging.)

Returns

0 for success, <0 on error

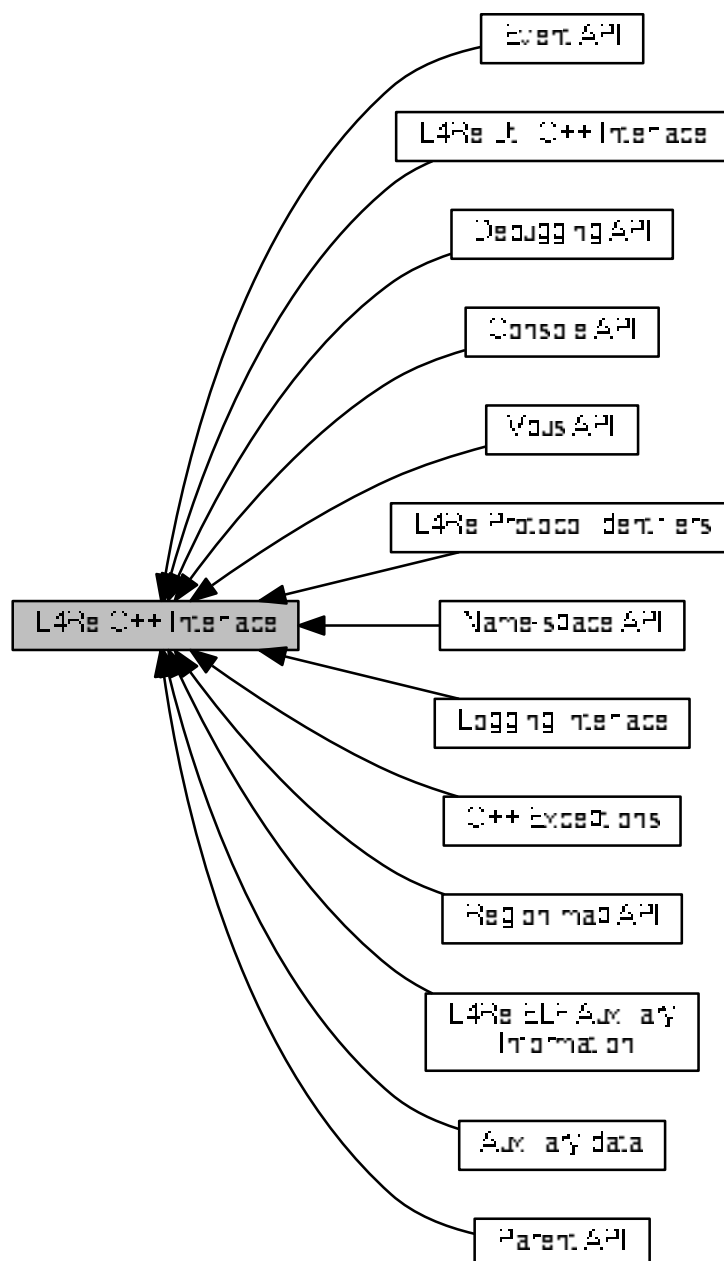
See also

[L4Re::Inhibitor::acquire\(\)](#)

12.66 L4Re C++ Interface

Documentation of the [L4](#) Runtime Environment C++ API.

Collaboration diagram for L4Re C++ Interface:



Modules

- [Auxiliary data](#)

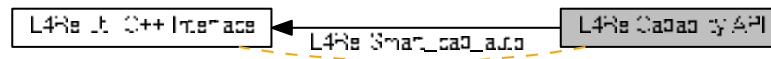
- [C++ Exceptions](#)
- [Console API](#)
Console interface.
- [Debugging API](#)
Debugging Interface.
- [Event API](#)
Event API.
- [L4Re ELF Auxiliary Information](#)
API for embedding auxiliary information into binary programs.
- [L4Re Protocol identifiers](#)
Fix L4Re Protocol Constants.
- [L4Re Util C++ Interface](#)
Documentation of the L4 Runtime Environment utility functionality in C++.
- [Logging interface](#)
Interface for log output.
- [Name-space API](#)
API for name spaces that store capabilities.
- [Parent API](#)
Parent interface.
- [Region map API](#)
Virtual address-space management.
- [Vbus API](#)
C++ interface of the Vbus API.

12.66.1 Detailed Description

Documentation of the [L4](#) Runtime Environment C++ API.

12.67 L4Re Capability API

Collaboration diagram for L4Re Capability API:



Data Structures

- class [L4Re::Cap_alloc](#)
Capability allocator interface.
- class [L4Re::Smart_cap_auto< Unmap_flags >](#)
Helper for Auto_cap and Auto_del_cap.
- class [L4Re::Smart_count_cap< Unmap_flags >](#)
Helper for Ref_cap and Ref_del_cap.
- class [L4Re::Util::Smart_cap_auto< Unmap_flags >](#)
Helper for Auto_cap and Auto_del_cap.
- class [L4Re::Util::Smart_count_cap< Unmap_flags >](#)
Helper for Ref_cap and Ref_del_cap.
- struct [L4Re::Util::Auto_cap< T >](#)
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [L4Re::Util::Auto_del_cap< T >](#)
Automatic capability that implements automatic free and unmap+delete of the capability selector.
- struct [L4Re::Util::Ref_cap< T >](#)
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [L4Re::Util::Ref_del_cap< T >](#)
Automatic capability that implements automatic free and unmap+delete of the capability selector.

Functions

- template<typename T >
[Auto_cap< T >::Cap L4Re::Util::make_auto_cap \(\)](#)
Allocate a capability slot and wrap it in an [Auto_cap](#).
- template<typename T >
[Auto_del_cap< T >::Cap L4Re::Util::make_auto_del_cap \(\)](#)
Allocate a capability slot and wrap it in an [Auto_del_cap](#).
- template<typename T >
[Ref_cap< T >::Cap L4Re::Util::make_ref_cap \(\)](#)
Allocate a capability slot and wrap it in a [Ref_cap](#).
- template<typename T >
[Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap \(\)](#)
Allocate a capability slot and wrap it in a [Ref_del_cap](#).
- virtual [L4Re::Cap_alloc::~Cap_alloc \(\)=0](#)
Destructor.

Variables

- [_Cap_alloc](#) & [L4Re::Util::cap_alloc](#)
Capability allocator.

12.67.1 Detailed Description

12.67.2 Function Documentation

12.67.2.1 `make_auto_cap()`

```
template<typename T >
Auto\_cap<T>::Cap L4Re::Util::make_auto_cap ( )
```

Allocate a capability slot and wrap it in an [Auto_cap](#).

Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Deprecated Use [L4Re::Util::make_unique_cap\(\)](#).

Definition at line 293 of file [cap_alloc](#).

12.67.2.2 `make_auto_del_cap()`

```
template<typename T >
Auto\_del\_cap<T>::Cap L4Re::Util::make_auto_del_cap ( )
```

Allocate a capability slot and wrap it in an [Auto_del_cap](#).

Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Deprecated Use [L4Re::Util::make_unique_del_cap\(\)](#).

Definition at line 305 of file [cap_alloc](#).

12.67.2.3 make_ref_cap()

```
template<typename T >
Ref_cap<T>::Cap L4Re::Util::make_ref_cap ( )
```

Allocate a capability slot and wrap it in a [Ref_cap](#).

Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Definition at line 316 of file [cap_alloc](#).

12.67.2.4 make_ref_del_cap()

```
template<typename T >
Ref_del_cap<T>::Cap L4Re::Util::make_ref_del_cap ( )
```

Allocate a capability slot and wrap it in a [Ref_del_cap](#).

Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Definition at line 325 of file [cap_alloc](#).

12.67.3 Variable Documentation

12.67.3.1 cap_alloc

```
_Cap_alloc& L4Re::Util::cap_alloc
```

Capability allocator.

This is the instance of the capability allocator that is used by usual applications. The actual implementation of the allocator depends on the configuration of the system.

Per default we use [Counting_cap_alloc](#), a reference-counting capability allocator, that keeps a reference counter for each managed capability selector.

Note

This capability allocator is not thread-safe.

Examples:

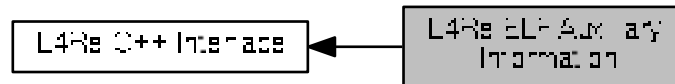
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), [examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Referenced by [L4Re::Util::make_shared_cap\(\)](#), [L4Re::Util::make_shared_del_cap\(\)](#), [L4Re::Util::make_unique_cap\(\)](#), [L4Re::Util::make_unique_del_cap\(\)](#), and [L4Re::Util::Object_registry::unregister_obj\(\)](#).

12.68 L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

Collaboration diagram for L4Re ELF Auxiliary Information:



Data Structures

- struct [l4re_elf_aux_t](#)
Generic header for each auxiliary vector element.
- struct [l4re_elf_aux_vma_t](#)
Auxiliary vector element for a reserved virtual memory area.
- struct [l4re_elf_aux_mword_t](#)
Auxiliary vector element for a single unsigned data word.

Macros

- `#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"), aligned(sizeof(l4re_elf_aux_mword_t))))`
Define an auxiliary vector element.
- `#define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}`
Define an auxiliary vector element.

Typedefs

- `typedef struct l4re_elf_aux_t l4re_elf_aux_t`
Generic header for each auxiliary vector element.
- `typedef struct l4re_elf_aux_vma_t l4re_elf_aux_vma_t`
Auxiliary vector element for a reserved virtual memory area.
- `typedef struct l4re_elf_aux_mword_t l4re_elf_aux_mword_t`
Auxiliary vector element for a single unsigned data word.

Enumerations

- `enum {
L4RE_ELF_AUX_T_NONE = 0, L4RE_ELF_AUX_T_VMA, L4RE_ELF_AUX_T_STACK_SIZE, L4RE_ELF_AUX_T_STACK_ADDR,
L4RE_ELF_AUX_T_KIP_ADDR }`

12.68.1 Detailed Description

API for embedding auxiliary information into binary programs.

This API allows information for the binary loader to be embedded into a binary application. This information can be reserved areas in the virtual memory of an application and things such as the stack size to be allocated for the first application thread.

12.68.2 Macro Definition Documentation

12.68.2.1 L4RE_ELF_AUX_ELEM

```
#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro14re_elf_aux"), aligned(sizeof(l4re_elf_aux_vma_t))))
```

Define an auxiliary vector element.

This is the generic method for defining auxiliary vector elements. A more convenient way is to use `L4RE_ELF_AUX_ELEM_T`.

Usage:

```
L4RE_ELF_AUX_ELEM l4re_elf_aux_vma_t decl_name =
{ L4RE_ELF_AUX_T_VMA, sizeof(l4re_elf_aux_vma_t), 0x2000, 0x4000 };
```

Definition at line 52 of file [elf_aux.h](#).

12.68.2.2 L4RE_ELF_AUX_ELEM_T

```
#define L4RE_ELF_AUX_ELEM_T(
    type,
    id,
    tag,
    val... ) static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
```

Define an auxiliary vector element.

Parameters

<i>type</i>	is the data type for the element (e.g., l4re_elf_aux_vma_t)
<i>id</i>	is the identifier (variable name) for the declaration (the variable is defined with <code>static</code> storage class)
<i>tag</i>	is the tag value for the element e.g., L4RE_ELF_AUX_T_VMA
<i>val</i>	are the values to be set in the descriptor

Usage:

```
L4RE_ELF_AUX_ELEM_T(l4re_elf_aux_vma_t, decl_name,  
    L4RE_ELF_AUX_T_VMA, 0x2000, 0x4000 );
```

Definition at line 67 of file [elf_aux.h](#).

12.68.3 Enumeration Type Documentation

12.68.3.1 anonymous enum

anonymous enum

Enumerator

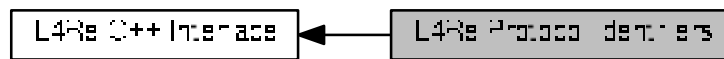
L4RE_ELF_AUX_T_NONE	Tag for an invalid element in the auxiliary vector.
L4RE_ELF_AUX_T_VMA	Tag for descriptor for a reserved virtual memory area.
L4RE_ELF_AUX_T_STACK_SIZE	Tag for descriptor that defines the stack size for the first application thread.
L4RE_ELF_AUX_T_STACK_ADDR	Tag for descriptor that defines the stack address for the first application thread.
L4RE_ELF_AUX_T_KIP_ADDR	Tag for descriptor that defines the KIP address for the binaries address space.

Definition at line 70 of file [elf_aux.h](#).

12.69 L4Re Protocol identifiers

Fix [L4Re](#) Protocol Constants.

Collaboration diagram for L4Re Protocol identifiers:



Enumerations

- enum [L4Re::Dataspace_::Opcodes](#)
Data-space communication-protocol opcodes.
- enum [L4Re::Event_::Opcodes](#)
Event communication-protocol opcodes.
- enum [L4Re::Inhibitor_::Opcodes](#)
Inhibitor communication-protocol opcodes.
- enum [L4Re::Log_::Opcodes](#)
Logging-service communication-protocol opcodes.
- enum [L4Re::Mem_alloc_::Opcodes](#)
Memory-allocator communication-protocol opcodes.
- enum [L4Re::Namespace_::Opcodes](#)
Name-space communication-protocol opcodes.
- enum [L4Re::Parent_::Opcodes](#)
Parent communication-protocol opcodes.
- enum [L4Re::Rm_::Opcodes](#)
Region-map communication-protocol opcodes.
- enum [L4Re::Video::Goos_::Opcodes](#)
Frame buffer communication-protocol opcodes.

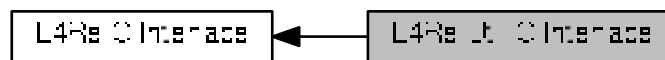
12.69.1 Detailed Description

Fix [L4Re](#) Protocol Constants.

12.70 L4Re Util C Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C.

Collaboration diagram for L4Re Util C Interface:



Documentation of the [L4](#) Runtime Environment utility functionality in C.

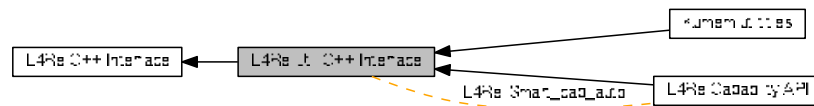
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

12.71 L4Re Util C++ Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Collaboration diagram for L4Re Util C++ Interface:



Modules

- [Kumem utilities](#)
- [L4Re Capability API](#)

Data Structures

- class [L4Re::Smart_cap_auto< Unmap_flags >](#)
Helper for Auto_cap and Auto_del_cap.
- class [L4Re::Util::Cap_alloc_base](#)
Capability allocator.
- class [L4Re::Util::Br_manager](#)
Buffer-register (BR) manager for [L4::Server](#).
- class [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >](#)
Internal reference-counting cap allocator.
- class [L4Re::Util::Event_buffer_t< PAYLOAD >](#)
Event_buffer utility class.
- class [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >](#)
An event buffer consumer.
- class [L4Re::Util::Vcon_svr< SVR >](#)
[Console](#) server template class.
- class [L4Re::Util::Video::Goos_svr](#)
Goos server class.

12.71.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

12.72 L4vbus GPIO functions

Collaboration diagram for L4vbus GPIO functions:



Data Structures

- class `L4vbus::Gpio_pin`
A *GPIO pin*.
- class `L4vbus::Gpio_module`
A *Gpio_module* groups multiple *GPIO pins* together.

Enumerations

- enum `L4vbus_gpio_generic_func` { `L4VBUS_GPIO_SETUP_INPUT` = 0x100, `L4VBUS_GPIO_SETUP_OUTPUT` = 0x200, `L4VBUS_GPIO_SETUP_IRQ` = 0x300 }
 - enum `L4vbus_gpio_pull_modes` { `L4VBUS_GPIO_PIN_PULL_NONE` = 0x100, `L4VBUS_GPIO_PIN_PULL_UP` = 0x200, `L4VBUS_GPIO_PIN_PULL_DOWN` = 0x300 }
- Constants for generic *GPIO* functions.
- Constants for generic *GPIO* pull up/down resistor configuration.

Functions

- int `l4vbus_gpio_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode, int value)
Configure the function of a *GPIO pin*.
- int `l4vbus_gpio_config_pull` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode)
Generic function to set pull up/down mode.
- int `l4vbus_gpio_config_pad` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned value)
Hardware specific configuration function.
- int `l4vbus_gpio_config_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned *value)
Read hardware specific configuration.
- int `l4vbus_gpio_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin)
Read value of *GPIO* input pin.
- int `l4vbus_gpio_set` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, int value)
Set *GPIO* output pin.
- int `l4vbus_gpio_multi_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned mode, unsigned value)

Configure function of multiple GPIO pins at once.

- int [l4vbus_gpio_multi_config_pad](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) handle, unsigned offset, unsigned mask, unsigned func, unsigned value)

Hardware specific configuration function for multiple GPIO pins.

- int [l4vbus_gpio_multi_get](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) handle, unsigned offset, unsigned *data)

Read values of multiple GPIO pins at once.

- int [l4vbus_gpio_multi_set](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) handle, unsigned offset, unsigned mask, unsigned data)

Set multiple GPIO output pins at once.

- int [l4vbus_gpio_to_irq](#) ([l4_cap_idx_t](#) vbus, [l4vbus_device_handle_t](#) handle, unsigned pin)

Create IRQ for GPIO pin.

12.72.1 Detailed Description

12.72.2 Enumeration Type Documentation

12.72.2.1 L4vbus_gpio_generic_func

enum [L4vbus_gpio_generic_func](#)

Constants for generic GPIO functions.

Enumerator

L4VBUS_GPIO_SETUP_INPUT	Set GPIO pin to input.
L4VBUS_GPIO_SETUP_OUTPUT	Set GPIO pin to output.
L4VBUS_GPIO_SETUP_IRQ	Set GPIO pin to IRQ.

Definition at line 26 of file [vbus_gpio.h](#).

12.72.2.2 L4vbus_gpio_pull_modes

enum [L4vbus_gpio_pull_modes](#)

Constants for generic GPIO pull up/down resistor configuration.

Enumerator

L4VBUS_GPIO_PIN_PULL_NONE	No pull up or pull down resistors.
L4VBUS_GPIO_PIN_PULL_UP	enable pull up resistor
L4VBUS_GPIO_PIN_PULL_DOWN	enable pull down resistor

Definition at line 36 of file [vbus_gpio.h](#).

12.72.3 Function Documentation

12.72.3.1 l4vbus_gpio_config_get()

```
int l4vbus_gpio_config_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned func,
    unsigned * value )
```

Read hardware specific configuration.

Parameters

	<i>vbus</i>	V-BUS capability
	<i>handle</i>	Device handle for the GPIO chip
	<i>pin</i>	GPIO pin number
	<i>func</i>	Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address.
out	<i>value</i>	The configuration value.

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_pin::config_get\(\)](#).

Here is the caller graph for this function:



12.72.3.2 l4vbus_gpio_config_pad()

```
int l4vbus_gpio_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned func,
    unsigned value )
```

Hardware specific configuration function.

Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pin

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_pin::config_pad\(\)](#).

Here is the caller graph for this function:



12.72.3.3 l4vbus_gpio_config_pull()

```
int l4vbus_gpio_config_pull (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned mode )
```

Generic function to set pull up/down mode.

Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>mode</i>	mode for pull up/down resistors, see L4vbus_gpio_pull_modes

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_pin::config_pull\(\)](#).

Here is the caller graph for this function:

**12.72.3.4 l4vbus_gpio_get()**

```
int l4vbus_gpio_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin )
```

Read value of GPIO input pin.

Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number to read from

Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Referenced by [L4vbus::Gpio_pin::get\(\)](#).

Here is the caller graph for this function:



12.72.3.5 l4vbus_gpio_multi_config_pad()

```
int l4vbus_gpio_multi_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned func,
    unsigned value )
```

Hardware specific configuration function for multiple GPIO pins.

Parameters

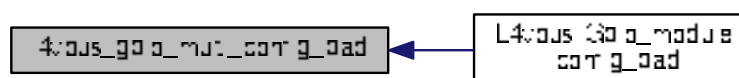
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address.
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pins

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_module::config_pad\(\)](#).

Here is the caller graph for this function:



12.72.3.6 l4vbus_gpio_multi_get()

```
int l4vbus_gpio_multi_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned * data )
```

Read values of multiple GPIO pins at once.

Parameters

	<i>vbus</i>	V-BUS capability
	<i>handle</i>	Device handle for the GPIO chip
	<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
out	<i>data</i>	Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined.

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_module::get\(\)](#).

Here is the caller graph for this function:



12.72.3.7 l4vbus_gpio_multi_set()

```

int l4vbus_gpio_multi_set (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned data )
  
```

Set multiple GPIO output pins at once.

Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation.
<i>data</i>	Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set.

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_module::set\(\)](#).

Here is the caller graph for this function:

**12.72.3.8 l4vbus_gpio_multi_setup()**

```

int l4vbus_gpio_multi_setup (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned mode,
    unsigned value )
  
```

Configure function of multiple GPIO pins at once.

Parameters

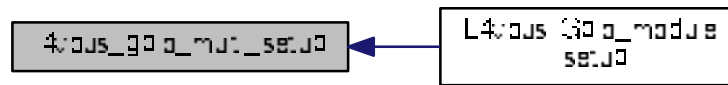
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mode</i>	GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pins to if they are configured as output pins

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_module::setup\(\)](#).

Here is the caller graph for this function:



12.72.3.9 l4vbus_gpio_set()

```
int l4vbus_gpio_set (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    int value )
```

Set GPIO output pin.

Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number to write to
<i>value</i>	Value to write to the GPIO pin (usually 0 or 1)

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_pin::set\(\)](#).

Here is the caller graph for this function:



12.72.3.10 l4vbus_gpio_setup()

```
int l4vbus_gpio_setup (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned mode,
    int value )
```

Configure the function of a GPIO pin.

Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>mode</i>	GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pin to if it is configured as an output pin

Returns

0 if OK, error code otherwise

Referenced by [L4vbus::Gpio_pin::setup\(\)](#).

Here is the caller graph for this function:



12.72.3.11 l4vbus_gpio_to_irq()

```
int l4vbus_gpio_to_irq (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin )
```

Create IRQ for GPIO pin.

Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin to create an IRQ for.

Returns

IRQ number if OK, negative error code otherwise

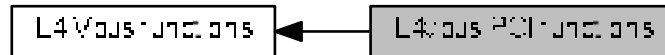
Referenced by [L4vbus::Gpio_pin::to_irq\(\)](#).

Here is the caller graph for this function:



12.73 L4vbus PCI functions

Collaboration diagram for L4vbus PCI functions:



Functions

- `int l4vbus_pci_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`
Read from the vPCI configuration space using the PCI root bridge.
- `int l4vbus_pci_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`
Write to the vPCI configuration space using the PCI root bridge.
- `int l4vbus_pci_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, int pin, unsigned char *trigger, unsigned char *polarity)`
Enable PCI interrupt for a specific device using the PCI root bridge.
- `int l4vbus_pcidv_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`
Read from the device's vPCI configuration space.
- `int l4vbus_pcidv_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`
Write to the device's vPCI configuration space.
- `int l4vbus_pcidv_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned char *trigger, unsigned char *polarity)`
Enable the device's PCI interrupt.

12.73.1 Detailed Description

12.73.2 Function Documentation

12.73.2.1 l4vbus_pci_cfg_read()

```

int l4vbus_pci_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width )
  
```

Read from the vPCI configuration space using the PCI root bridge.

Parameters

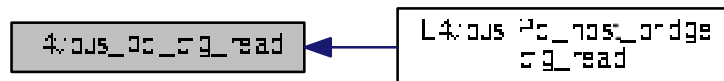
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by [L4vbus::Pci_host_bridge::cfg_read\(\)](#).

Here is the caller graph for this function:



12.73.2.2 l4vbus_pci_cfg_write()

```

int l4vbus_pci_cfg_write (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t value,
    l4_uint32_t width )
  
```

Write to the vPCI configuration space using the PCI root bridge.

Parameters

<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI root bridge
<i>bus</i>	Bus number
<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by [L4vbus::Pci_host_bridge::cfg_write\(\)](#).

Here is the caller graph for this function:

**12.73.2.3 l4vbus_pci_irq_enable()**

```

int l4vbus_pci_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    int pin,
    unsigned char * trigger,
    unsigned char * polarity )
  
```

Enable PCI interrupt for a specific device using the PCI root bridge.

Parameters

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>pin</i>	Interrupt pin (normally as reported in configuration register INTR)
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

Referenced by [L4vbus::Pci_host_bridge::irq_enable\(\)](#).

Here is the caller graph for this function:



12.73.2.4 l4vbus_pci_dev_cfg_read()

```

int l4vbus_pci_dev_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width )
  
```

Read from the device's vPCI configuration space.

Parameters

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by [L4vbus::Pci_dev::cfg_read\(\)](#).

Here is the caller graph for this function:



12.73.2.5 l4vbus_pcidev_cfg_write()

```
int l4vbus_pcidev_cfg_write (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t reg,
    l4_uint32_t value,
    l4_uint32_t width )
```

Write to the device's vPCI configuration space.

Parameters

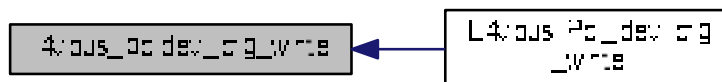
<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI device
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by [L4vbus::Pci_dev::cfg_write\(\)](#).

Here is the caller graph for this function:



12.73.2.6 l4vbus_pcidev_irq_enable()

```
int l4vbus_pcidev_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned char * trigger,
    unsigned char * polarity )
```

Enable the device's PCI interrupt.

Parameters

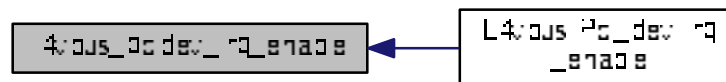
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

Referenced by [L4vbus::Pci_dev::irq_enable\(\)](#).

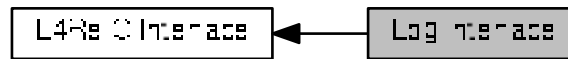
Here is the caller graph for this function:



12.74 Log interface

Log C interface.

Collaboration diagram for Log interface:



Functions

- void [l4re_log_print](#) (char const *string) [L4_NOTHROW](#)
Write a null terminated string to the default log.
- void [l4re_log_printn](#) (char const *string, int len) [L4_NOTHROW](#)
Write a string of a given length to the default log.
- void [l4re_log_print_srv](#) (const [l4_cap_idx_t](#) logcap, char const *string) [L4_NOTHROW](#)
Write a null terminated string to a log.
- void [l4re_log_printn_srv](#) (const [l4_cap_idx_t](#) logcap, char const *string, int len) [L4_NOTHROW](#)
Write a string of a given length to a log.

12.74.1 Detailed Description

Log C interface.

12.74.2 Function Documentation

12.74.2.1 l4re_log_print()

```
void l4re_log_print (
    char const * string ) [inline]
```

Write a null terminated string to the default log.

Parameters

<i>string</i>	Text to print, null terminated.
---------------	---------------------------------

Returns

0 for success, <0 on error

See also

[L4Re::Log::print](#)

Definition at line 99 of file [log.h](#).

References [l4re_log_print_srv\(\)](#).

Here is the call graph for this function:

**12.74.2.2 l4re_log_print_srv()**

```
void l4re_log_print_srv (  
    const l4\_cap\_idx\_t logcap,  
    char const * string )
```

Write a null terminated string to a log.

Parameters

<i>logcap</i>	Log capability (service).
<i>string</i>	Text to print, null terminated.

Returns

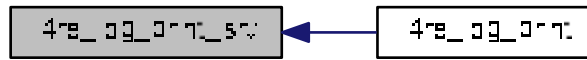
0 for success, <0 on error

See also

[L4Re::Log::print](#)

Referenced by [l4re_log_print\(\)](#).

Here is the caller graph for this function:



12.74.2.3 l4re_log_printn()

```
void l4re_log_printn (
    char const * string,
    int len ) [inline]
```

Write a string of a given length to the default log.

Parameters

<i>string</i>	Text to print, null terminated.
<i>len</i>	Length of string in bytes.

Returns

0 for success, <0 on error

See also

[L4Re::Log::println](#)

Definition at line 105 of file [log.h](#).

References [l4re_log_printn_srv\(\)](#).

Here is the call graph for this function:



12.74.2.4 l4re_log_printn_srv()

```
void l4re_log_printn_srv (
    const l4_cap_idx_t logcap,
    char const * string,
    int len )
```

Write a string of a given length to a log.

Parameters

<i>logcap</i>	Log capability (service).
<i>string</i>	Text to print, null terminated.
<i>len</i>	Length of string in bytes.

Returns

0 for success, <0 on error

See also

[L4Re::Log::println](#)

Referenced by [l4re_log_printn\(\)](#).

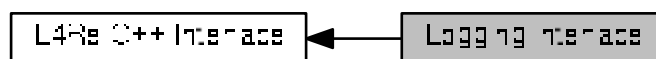
Here is the caller graph for this function:



12.75 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



Data Structures

- class [L4Re::Log](#)
Log interface class.

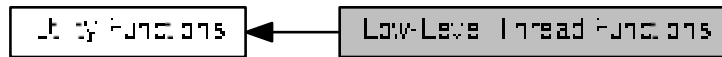
12.75.1 Detailed Description

Interface for log output.

The logging interface provides a facility sending log output. One purpose of the interface is to serialize the output and provide the possibility to tag output sent to a specific log object.

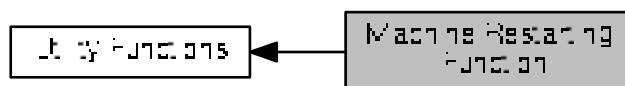
12.76 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



12.77 Machine Restarting Function

Collaboration diagram for Machine Restarting Function:



Functions

- void [l4util_reboot](#) (void)
Machine reboot.

12.77.1 Detailed Description

12.78 Memory allocator

Memory allocator C interface.

Collaboration diagram for Memory allocator:



Enumerations

- enum [l4re_ma_flags](#)
Flags for requesting memory at the memory allocator.

Functions

- long [l4re_ma_alloc](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags) [L4_NOTHROW](#)
Allocate memory.
- long [l4re_ma_alloc_align](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags, unsigned long align) [L4_NOTHROW](#)
Allocate memory.
- long [l4re_ma_free](#) ([l4re_ds_t](#) const mem) [L4_NOTHROW](#)
Free memory.
- long [l4re_ma_alloc_align_srv](#) ([l4_cap_idx_t](#) srv, long size, [l4re_ds_t](#) const mem, unsigned long flags, unsigned long align) [L4_NOTHROW](#)
Allocate memory.
- long [l4re_ma_free_srv](#) ([l4_cap_idx_t](#) srv, [l4re_ds_t](#) const mem) [L4_NOTHROW](#)
Free memory.

12.78.1 Detailed Description

Memory allocator C interface.

12.78.2 Enumeration Type Documentation

12.78.2.1 l4re_ma_flags

enum [l4re_ma_flags](#)

Flags for requesting memory at the memory allocator.

See also

[L4Re::Mem_alloc::Mem_alloc_flags](#)

Definition at line 42 of file [mem_alloc.h](#).

12.78.3 Function Documentation

12.78.3.1 l4re_ma_alloc()

```
long l4re_ma_alloc (
    long size,
    l4re_ds_t const mem,
    unsigned long flags ) [inline]
```

Allocate memory.

Parameters

<i>size</i>	Size to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).
<i>mem</i>	Capability slot to put the requested dataspace in
<i>flags</i>	Flags, see l4re_ma_flags

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::alloc](#)

The memory allocator returns a dataspace.

Note

This function is using the [L4Re::Env::env\(\)](#)->mem_alloc() service.

Examples:

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 167 of file [mem_alloc.h](#).

References [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:



12.78.3.2 l4re_ma_alloc_align()

```

long l4re_ma_alloc_align (
    long size,
    l4re_ds_t const mem,
    unsigned long flags,
    unsigned long align ) [inline]
  
```

Allocate memory.

Parameters

<i>size</i>	Size to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).
<i>mem</i>	Capability slot to put the requested dataspace in
<i>flags</i>	Flags, see l4re_ma_flags
<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT, default 0

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::alloc](#) and
[l4re_ma_alloc](#)

The memory allocator returns a dataspace.

Note

This function is using the [L4Re::Env::env\(\)](#)->mem_alloc() service.

Definition at line 175 of file [mem_alloc.h](#).

References [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:

**12.78.3.3 l4re_ma_alloc_align_srv()**

```

long l4re_ma_alloc_align_srv (
    l4_cap_idx_t srv,
    long size,
    l4re_ds_t const mem,
    unsigned long flags,
    unsigned long align )
  
```

Allocate memory.

Parameters

<i>srv</i>	Memory allocator service.
<i>size</i>	Size to be requested.
<i>mem</i>	Capability slot to put the requested dataspace in
<i>flags</i>	Flags, see l4re_ma_flags
<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT, default 0

Returns

0 on success, <0 on error

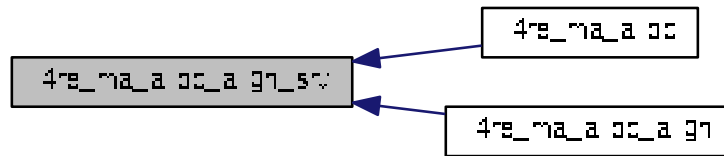
See also

[L4Re::Mem_alloc::alloc](#)

The memory allocator returns a dataspace.

Referenced by [l4re_ma_alloc\(\)](#), and [l4re_ma_alloc_align\(\)](#).

Here is the caller graph for this function:



12.78.3.4 l4re_ma_free()

```
long l4re_ma_free (
    l4re_ds_t const mem ) [inline]
```

Free memory.

Parameters

<i>mem</i>	Dataspace to free.
------------	--------------------

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::free](#)

Note

This function is using the [L4Re::Env::env\(\)](#)->[mem_alloc\(\)](#) service.

Deprecated This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Definition at line 183 of file [mem_alloc.h](#).

References [l4re_ma_free_srv\(\)](#).

Here is the call graph for this function:



12.78.3.5 l4re_ma_free_srv()

```

long l4re_ma_free_srv (
    l4_cap_idx_t srv,
    l4re_ds_t const mem )
  
```

Free memory.

Parameters

<i>srv</i>	Memory allocator service.
<i>mem</i>	Dataspace to free.

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::free](#)

Deprecated This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Referenced by [l4re_ma_free\(\)](#).

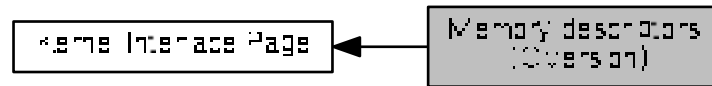
Here is the caller graph for this function:



12.79 Memory descriptors (C version)

C Interface for KIP memory descriptors.

Collaboration diagram for Memory descriptors (C version):



Data Structures

- struct `l4_kernel_info_mem_desc_t`
Memory descriptor data structure.

Typedefs

- typedef struct `l4_kernel_info_mem_desc_t` `l4_kernel_info_mem_desc_t`
Memory descriptor data structure.

Enumerations

- enum `l4_mem_type_t` {
`l4_mem_type_undefined` = 0x0, `l4_mem_type_conventional` = 0x1, `l4_mem_type_reserved` = 0x2, `l4_mem_type_dedicated` = 0x3,
`l4_mem_type_shared` = 0x4, `l4_mem_type_info` = 0xd, `l4_mem_type_bootloader` = 0xe, `l4_mem_type_archspecific` = 0xf }
Type of a memory descriptor.
- enum `l4_mem_info_sub_type_t` { `l4_mem_info_acpi_rsdp` = 0 }
Memory sub types for `l4_mem_type_info` descriptors.

Functions

- `l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`
Get pointer to memory descriptors from KIP.
- `unsigned l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`
Get number of memory descriptors in KIP.
- `void l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW`
Populate a memory descriptor.
- `l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`
Get start address of the region described by the memory descriptor.
- `l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`

Get end address of the region described by the memory descriptor.

- `l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`

Get type of the memory region.

- `l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`

Get sub-type of memory region.

- `l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`

Get virtual flag of the memory descriptor.

12.79.1 Detailed Description

C Interface for KIP memory descriptors.

Include File

```
#include <l4/sys/memdesc.h>
```

This module contains the C functions to access the memory descriptor in the kernel interface page (KIP).

12.79.2 Typedef Documentation

12.79.2.1 l4_kernel_info_mem_desc_t

```
typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t
```

Memory descriptor data structure.

Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

12.79.3 Enumeration Type Documentation

12.79.3.1 l4_mem_info_sub_type_t

```
enum l4_mem_info_sub_type_t
```

Memory sub types for l4_mem_type_info descriptors.

Enumerator

<code>l4_mem_info_acpi_rsdp</code>	Physical address of the ACPI root pointer.
------------------------------------	--

Definition at line 61 of file [memdesc.h](#).

12.79.3.2 l4_mem_type_t

enum [l4_mem_type_t](#)

Type of a memory descriptor.

Enumerator

l4_mem_type_undefined	Undefined, unused descriptor.
l4_mem_type_conventional	Conventional memory.
l4_mem_type_reserved	Reserved memory for kernel etc.
l4_mem_type_dedicated	Dedicated memory (some device memory)
l4_mem_type_shared	Shared memory (not implemented)
l4_mem_type_info	Info from the boot loader.
l4_mem_type_bootloader	Memory owned by the boot loader.
l4_mem_type_archspecific	Architecture specific memory (e.g., ACPI memory)

Definition at line 44 of file [memdesc.h](#).

12.79.4 Function Documentation

12.79.4.1 l4_kernel_info_get_mem_desc_end()

```
l4_umword_t l4_kernel_info_get_mem_desc_end (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get end address of the region described by the memory descriptor.

Returns

End address.

Definition at line 217 of file [memdesc.h](#).

12.79.4.2 l4_kernel_info_get_mem_desc_is_virtual()

```
l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get virtual flag of the memory descriptor.

Returns

1 if region is virtual memory, 0 if region is physical memory

Definition at line 238 of file [memdesc.h](#).

12.79.4.3 l4_kernel_info_get_mem_desc_start()

```
l4_umword_t l4_kernel_info_get_mem_desc_start (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get start address of the region described by the memory descriptor.

Returns

Start address.

Definition at line 210 of file [memdesc.h](#).

12.79.4.4 l4_kernel_info_get_mem_desc_subtype()

```
l4_umword_t l4_kernel_info_get_mem_desc_subtype (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get sub-type of memory region.

Returns

Sub-type.

The sub type is defined for architecture specific memory descriptors (see [l4_mem_type_archspecific](#)) and has architecture specific meaning.

Definition at line 231 of file [memdesc.h](#).

12.79.4.5 l4_kernel_info_get_mem_desc_type()

```
l4_umword_t l4_kernel_info_get_mem_desc_type (
    l4_kernel_info_mem_desc_t * md ) [inline]
```

Get type of the memory region.

Returns

Type of the region (see [l4_mem_type_t](#)).

Definition at line 224 of file [memdesc.h](#).

12.79.4.6 l4_kernel_info_get_num_mem_descs()

```
unsigned l4_kernel_info_get_num_mem_descs (
    l4_kernel_info_t * kip ) [inline]
```

Get number of memory descriptors in KIP.

Returns

Number of memory descriptors.

Definition at line 188 of file [memdesc.h](#).

References [l4_kernel_info_t::mem_info](#).

12.79.4.7 l4_kernel_info_set_mem_desc()

```
void l4_kernel_info_set_mem_desc (
    l4_kernel_info_mem_desc_t * md,
    l4_addr_t start,
    l4_addr_t end,
    unsigned type,
    unsigned virt,
    unsigned sub_type ) [inline]
```

Populate a memory descriptor.

Parameters

<i>md</i>	Pointer to memory descriptor
<i>start</i>	Start of region
<i>end</i>	End of region
<i>type</i>	Type of region
<i>virt</i>	1 if virtual region, 0 if physical region
<i>sub_type</i>	Sub type.

Definition at line 195 of file [memdesc.h](#).

12.80 Memory operations.

Operations for memory access.

Collaboration diagram for Memory operations.:



Enumerations

- enum `L4_mem_op_widths` { `L4_MEM_WIDTH_1BYTE` = 0, `L4_MEM_WIDTH_2BYTE` = 1, `L4_MEM_WIDTH_4BYTE` = 2 }

Memory access width definitions.

Functions

- unsigned long `l4_mem_read` (unsigned long virtaddress, unsigned width)
Read user task memory from kernel privilege level.
- void `l4_mem_write` (unsigned long virtaddress, unsigned width, unsigned long value)
Write user task memory from kernel privilege level.

12.80.1 Detailed Description

Operations for memory access.

This module provides functionality to access user task memory from the kernel. This is needed for some devices that are only accessible from privileged processor mode. Only use this when absolutely required. This functionality is only available on the ARM architecture.

```
#include <l4/sys/mem_op.h>
```

12.80.2 Enumeration Type Documentation

12.80.2.1 L4_mem_op_widths

```
enum L4_mem_op_widths
```

Memory access width definitions.

Enumerator

L4_MEM_WIDTH_1BYTE	Access one byte (8-bit width)
L4_MEM_WIDTH_2BYTE	Access two bytes (16-bit width)
L4_MEM_WIDTH_4BYTE	Access four bytes (32-bit width)

Definition at line 51 of file [mem_op.h](#).

12.80.3 Function Documentation

12.80.3.1 l4_mem_read()

```
unsigned long l4_mem_read (  
    unsigned long virtaddress,  
    unsigned width ) [inline]
```

Read user task memory from kernel privilege level.

Parameters

<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2,

See also

[L4_mem_op_widths](#)

Returns

Read value.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 141 of file [mem_op.h](#).

References [l4_mem_arm_op_call\(\)](#).

Here is the call graph for this function:



12.80.3.2 l4_mem_write()

```
void l4_mem_write (
    unsigned long virtaddress,
    unsigned width,
    unsigned long value ) [inline]
```

Write user task memory from kernel privilege level.

Parameters

<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)
<i>value</i>	Value to write.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line [147](#) of file [mem_op.h](#).

References [l4_mem_arm_op_call\(\)](#).

Here is the call graph for this function:



12.81 Memory related

Memory related constants, data types and functions.

Collaboration diagram for Memory related:



Macros

- `#define L4_PAGESIZE`
Minimal page size (in bytes).
- `#define L4_PAGEMASK`
Mask for the page number.
- `#define L4_LOG2_PAGESIZE`
Number of bits used for page offset.
- `#define L4_SUPERPAGESIZE`
Size of a large page.
- `#define L4_SUPERPAGEMASK`
Mask for the number of a large page.
- `#define L4_LOG2_SUPERPAGESIZE`
Number of bits used as offset for a large page.
- `#define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)`
Invalid address as pointer type.
- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.
- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.
- `#define L4_PAGESHIFT 12`
Size of a page log2-based.
- `#define L4_SUPERPAGESHIFT 22`
Size of a large page log2-based.

Enumerations

- `enum l4_addr_consts_t { L4_INVALID_ADDR = ~0UL }`
Address related constants.

Functions

- [l4_addr_t l4_trunc_page \(l4_addr_t address\) L4_NOTHROW](#)
Round an address down to the next lower page boundary.
- [l4_addr_t l4_trunc_size \(l4_addr_t address, unsigned char bits\) L4_NOTHROW](#)
Round an address down to the next lower flex page with size bits.
- [l4_addr_t l4_round_page \(l4_addr_t address\) L4_NOTHROW](#)
Round address up to the next page.
- [l4_addr_t l4_round_size \(l4_umword_t value, unsigned char bits\) L4_NOTHROW](#)
Round value up to the next alignment with bits size.

12.81.1 Detailed Description

Memory related constants, data types and functions.

12.81.2 Macro Definition Documentation

12.81.2.1 L4_LOG2_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 325 of file [consts.h](#).

Referenced by [L4Re::Rm::attach\(\)](#), [L4Re::Dataspace::map\(\)](#), [L4Re::Dataspace::map_region\(\)](#), and [L4Re::Util::↵
Dataspace_srv::page_shift\(\)](#).

12.81.2.2 L4_LOG2_SUPERPAGESIZE

```
#define L4_LOG2_SUPERPAGESIZE
```

Number of bits used as offset for a large page.

Size of large page in log2

Definition at line 351 of file [consts.h](#).

12.81.2.3 L4_PAGEMASK

```
#define L4_PAGEMASK
```

Mask for the page number.

Note

The most significant bits are set.

Definition at line 316 of file [consts.h](#).

Referenced by [l4_round_page\(\)](#), [l4_sleep_forever\(\)](#), and [l4_trunc_page\(\)](#).

12.81.2.4 L4_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

Note

The most significant bits are set.

Definition at line 343 of file [consts.h](#).

12.81.2.5 L4_SUPERPAGESIZE

```
#define L4_SUPERPAGESIZE
```

Size of a large page.

A large page is a *super page* on IA32 or a *section* on ARM.

Definition at line 334 of file [consts.h](#).

12.81.3 Enumeration Type Documentation

12.81.3.1 l4_addr_consts_t

```
enum l4_addr_consts_t
```

Address related constants.

Enumerator

L4_INVALID_ADDR	Invalid address.
-----------------	------------------

Definition at line 407 of file [consts.h](#).

12.81.4 Function Documentation

12.81.4.1 l4_round_page()

```
l4_addr_t l4_round_page (
    l4_addr_t address ) [inline]
```

Round address up to the next page.

The address is rounded up to the next minimal page boundary. On most architectures this is a 4k page. Check [L4_PAGESIZE](#) for the minimal page size.

Parameters

<i>address</i>	The address to round up.
----------------	--------------------------

Definition at line 389 of file [consts.h](#).

References [L4_NOTHROW](#), [L4_PAGEMASK](#), [L4_PAGESIZE](#), and [l4_round_size\(\)](#).

Referenced by [L4Re::Rm::attach\(\)](#), [l4_trunc_size\(\)](#), [L4Re::Dataspace::map\(\)](#), and [L4Re::Dataspace::map_region\(\)](#).

Here is the call graph for this function:



[illegible]

```
l4_addr_t l4_round_size (
    l4_umword_t value,
    unsigned char bits ) [inline]
```

Parameters

<i>value</i>	The value to round up to the next size-alignment.
<i>bits</i>	The size of the alignment (log2).

Referenced by `cxn::List_alloc::alloc_max()`, `L4Re::Util::Dataspace_svr::is_static()`, `l4_round_page()`, `L4Re::Dataspace::map_region()`, and `L4virtio::Virtqueue::setup_simple()`.

[illegible]

12.81.4.3 l4_trunc_page()

```
l4_addr_t l4_trunc_page (
    l4_addr_t address ) [inline]
```

Round an address down to the next lower page boundary.

The address is rounded down to the next lower minimal page boundary. On most architectures this is a 4k page. Check [L4_PAGESIZE](#) for the minimal page size.

Parameters

<i>address</i>	The address to round.
----------------	-----------------------

Examples:

[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 364 of file [consts.h](#).

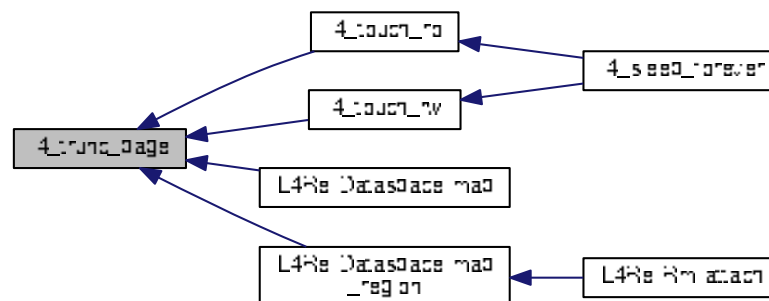
References [L4_NOTHROW](#), [L4_PAGEMASK](#), and [l4_trunc_size\(\)](#).

Referenced by [l4_touch_ro\(\)](#), [l4_touch_rw\(\)](#), [L4Re::Dataspace::map\(\)](#), and [L4Re::Dataspace::map_region\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.81.4.4 l4_trunc_size()

```
l4_addr_t l4_trunc_size (
    l4_addr_t address,
    unsigned char bits ) [inline]
```

Round an address down to the next lower flex page with size *bits*.

Parameters

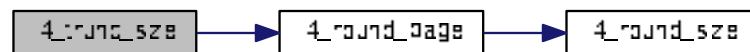
<i>address</i>	The address to round.
<i>bits</i>	The size of the flex page (log2).

Definition at line 375 of file consts.h.

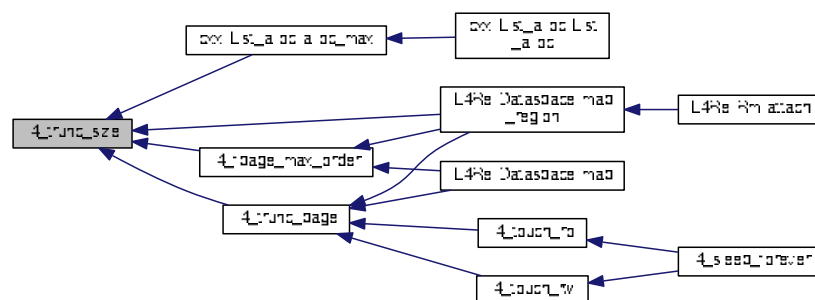
References [L4_NOTHROW](#), and [l4_round_page\(\)](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [l4_fpage_max_order\(\)](#), [l4_trunc_page\(\)](#), and [L4Re::Dataspace::map←_region\(\)](#).

Here is the call graph for this function:



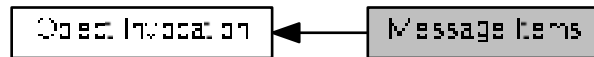
Here is the caller graph for this function:



12.82 Message Items

Message item related functions.

Collaboration diagram for Message Items:



Enumerations

- enum `l4_msg_item_consts_t` {
`L4_ITEM_MAP = 8, L4_ITEM_CONT = 1, L4_MAP_ITEM_GRANT = 2, L4_MAP_ITEM_MAP = 0,`
`L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2, L4_RCV_ITEM_LOCAL_ID = 4 }`

Constants for message items.

Functions

- `l4_umword_t l4_map_control (l4_umword_t spot, unsigned char cache, unsigned grant) L4_NOTHROW`
Create the first word for a map item for the memory space.
- `l4_umword_t l4_map_obj_control (l4_umword_t spot, unsigned grant) L4_NOTHROW`
Create the first word for a map item for the object space.

12.82.1 Detailed Description

Message item related functions.

Message items are typed items that can be transferred via IPC operations. Message items are also used to specify receive windows for typed items to be received. Message items are placed in the message registers (MRs) of the UTCB of the sending thread. Receive items are placed in the buffer registers (BRs) of the UTCB of the receiving thread.

Message items are usually two-word data structures. The first word denotes the type of the message item (for example a memory flex-page, io flex-page or object flex-page) and the second word contains information depending on the type. There is actually one exception that is a small (one word) receive buffer item for a single capability.

12.82.2 Enumeration Type Documentation

12.82.2.1 `l4_msg_item_consts_t`

```
enum l4_msg_item_consts_t
```

Constants for message items.

Enumerator

L4_ITEM_MAP	Identify a message item as <i>map item</i> .
L4_ITEM_CONT	Denote that the following item shall be put into the same receive item as this one.
L4_MAP_ITEM_GRANT	Flag as <i>grant</i> instead of <i>map</i> operation.
L4_MAP_ITEM_MAP	Flag as usual <i>map</i> operation.
L4_RCV_ITEM_SINGLE_CAP	Mark the receive buffer to be a small receive item that describes a buffer for a single capability.
L4_RCV_ITEM_LOCAL_ID	The receiver requests to receive a local ID instead of a mapping whenever possible.

Definition at line 187 of file [consts.h](#).

12.82.3 Function Documentation

12.82.3.1 l4_map_control()

```
l4_umword_t l4_map_control (
    l4_umword_t spot,
    unsigned char cache,
    unsigned grant ) [inline]
```

Create the first word for a map item for the memory space.

Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flex page have differing sizes.
<i>cache</i>	Cacheability hints for memory flex pages. See Cacheability options
<i>grant</i>	Indicates if it is a map or a grant item.

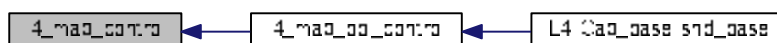
Returns

The value to be used as first word in a map item for memory.

Definition at line 672 of file [__l4_fpage.h](#).

Referenced by [l4_map_obj_control\(\)](#).

Here is the caller graph for this function:



12.82.3.2 l4_map_obj_control()

```
l4_umword_t l4_map_obj_control (
    l4_umword_t spot,
    unsigned grant ) [inline]
```

Create the first word for a map item for the object space.

Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flex pages have different size.
<i>grant</i>	Indicates if it is a map item or a grant item.

Returns

The value to be used as first word in a map item for kernel objects or IO-ports.

Definition at line 679 of file [__l4_fpage.h](#).

References [l4_map_control\(\)](#).

Referenced by [L4::Cap_base::snd_base\(\)](#).

Here is the call graph for this function:

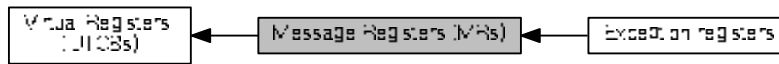


Here is the caller graph for this function:



12.83 Message Registers (MRs)

Collaboration diagram for Message Registers (MRs):



Modules

- [Exception registers](#)
Overly definition of the MRs for exception messages.

Data Structures

- [union l4_msg_regs_t](#)
Encapsulation of the message-register block in the UTCB.

Typedefs

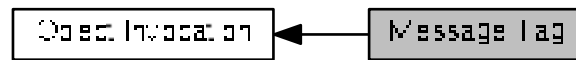
- `typedef union l4_msg_regs_t l4_msg_regs_t`
Encapsulation of the message-register block in the UTCB.

12.83.1 Detailed Description

12.84 Message Tag

API related to the message tag data type.

Collaboration diagram for Message Tag:



Data Structures

- struct [l4_msgtag_t](#)
Message tag data structure.

Typedefs

- typedef struct [l4_msgtag_t](#) [l4_msgtag_t](#)
Message tag data structure.

Enumerations

- enum [L4_platform_ctl_proto](#) { [L4_PROTO_PLATFORM_CTL](#) = 0 }
Predefined protocol type for messages to platform-control objects.
- enum [l4_msgtag_protocol](#) {
[L4_PROTO_NONE](#) = 0, [L4_PROTO_ALLOW_SYSCALL](#) = 1, [L4_PROTO_PF_EXCEPTION](#) = 1, [L4_PROTO_TO_IRQ](#) = -1L,
[L4_PROTO_PAGE_FAULT](#) = -2L, [L4_PROTO_PREEMPTION](#) = -3L, [L4_PROTO_SYS_EXCEPTION](#) = -4L,
[L4_PROTO_EXCEPTION](#) = -5L,
[L4_PROTO_SIGMA0](#) = -6L, [L4_PROTO_IO_PAGE_FAULT](#) = -8L, [L4_PROTO_KOBJECT](#) = -10L, [L4_PROTO_OTOTASK](#) = -11L,
[L4_PROTO_THREAD](#) = -12L, [L4_PROTO_LOG](#) = -13L, [L4_PROTO_SCHEDULER](#) = -14L, [L4_PROTO_FACTORY](#) = -15L,
[L4_PROTO_VM](#) = -16L, [L4_PROTO_DMA_SPACE](#) = -17L, [L4_PROTO_IRQ_SENDER](#) = -18L, [L4_PROTO_TO_IRQ_MUX](#) = -19L,
[L4_PROTO_SEMAPHORE](#) = -20L, [L4_PROTO_META](#) = -21L, [L4_PROTO_IOMMU](#) = -22L, [L4_PROTO_DEBUGGER](#) = -23L,
[L4_PROTO_SMCCC](#) = -24L }
Message tag for IPC operations.
- enum [l4_msgtag_flags](#) {
[L4_MSGTAG_ERROR](#), [L4_MSGTAG_TRANSFER_FPU](#), [L4_MSGTAG_SCHEDULE](#), [L4_MSGTAG_PROPOGATE](#),
[L4_MSGTAG_FLAGS](#) }
Flags for message tags.

Functions

- `l4_msgtag_t l4_msgtag` (long label, unsigned words, unsigned items, unsigned flags) `L4_NOTHROW`
Create a message tag from the specified values.
- long `l4_msgtag_label` (`l4_msgtag_t t`) `L4_NOTHROW`
Get the protocol of tag.
- unsigned `l4_msgtag_words` (`l4_msgtag_t t`) `L4_NOTHROW`
Get the number of untyped words.
- unsigned `l4_msgtag_items` (`l4_msgtag_t t`) `L4_NOTHROW`
Get the number of typed items.
- unsigned `l4_msgtag_flags` (`l4_msgtag_t t`) `L4_NOTHROW`
Get the flags.
- unsigned `l4_msgtag_has_error` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for error indicator flag.
- unsigned `l4_msgtag_is_page_fault` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for page-fault protocol.
- unsigned `l4_msgtag_is_preemption` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for preemption protocol.
- unsigned `l4_msgtag_is_sys_exception` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for system-exception protocol.
- unsigned `l4_msgtag_is_exception` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for exception protocol.
- unsigned `l4_msgtag_is_sigma0` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for sigma0 protocol.
- unsigned `l4_msgtag_is_io_page_fault` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for IO-page-fault protocol.

12.84.1 Detailed Description

API related to the message tag data type.

Include File

```
#include <l4/sys/types.h>
```

12.84.2 Typedef Documentation

12.84.2.1 `l4_msgtag_t`

```
typedef struct l4_msgtag_t l4_msgtag_t
```

Message tag data structure.

Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

12.84.3 Enumeration Type Documentation

12.84.3.1 l4_msgtag_flags

enum [l4_msgtag_flags](#)

Flags for message tags.

Enumerator

L4_MSGTAG_ERROR	Error indicator flag.
L4_MSGTAG_TRANSFER_FPU	Enable FPU transfer flag for IPC. By enabling this flag when sending IPC, the sender indicates that the contents of the FPU shall be transferred to the receiving thread. However, the receiver has to indicate its willingness to receive FPU context in its buffer descriptor register (BDR).
L4_MSGTAG_SCHEDULE	Enable schedule in IPC flag. Usually IPC operations donate the remaining time slice of a thread to the called thread. Enabling this flag when sending IPC does a real scheduling decision. However, this flag decreases IPC performance.
L4_MSGTAG_PROPAGATE	Enable IPC propagation. This flag enables IPC propagation, which means an IPC reply-connection from the current caller will be propagated to the new IPC receiver. This makes it possible to propagate an IPC call to a third thread, which may then directly answer to the caller.
L4_MSGTAG_FLAGS	Mask for all flags.

Definition at line 95 of file [types.h](#).

12.84.3.2 l4_msgtag_protocol

enum [l4_msgtag_protocol](#)

Message tag for IPC operations.

All predefined protocols used by the kernel.

Enumerator

L4_PROTO_NONE	Default protocol tag to reply to kernel.
L4_PROTO_ALLOW_SYSCALL	Allow an alien the system call.
L4_PROTO_PF_EXCEPTION	Make an exception out of a page fault.
L4_PROTO_IRQ	IRQ message.
L4_PROTO_PAGE_FAULT	Page fault message.
L4_PROTO_PREEMPTION	Preemption message.
L4_PROTO_SYS_EXCEPTION	System exception.
L4_PROTO_EXCEPTION	Exception.
L4_PROTO_SIGMA0	Sigma0 protocol.

Enumerator

L4_PROTO_IO_PAGE_FAULT	I/O page fault message.
L4_PROTO_KOBJECT	Protocol for messages to a generic kobject.
L4_PROTO_TASK	Protocol for messages to a task object.
L4_PROTO_THREAD	Protocol for messages to a thread object.
L4_PROTO_LOG	Protocol for messages to a log object.
L4_PROTO_SCHEDULER	Protocol for messages to a scheduler object.
L4_PROTO_FACTORY	Protocol for messages to a factory object.
L4_PROTO_VM	Protocol for messages to a virtual machine object.
L4_PROTO_DMA_SPACE	Protocol for (creating) kernel DMA space objects.
L4_PROTO_IRQ_SENDER	Protocol for IRQ senders (IRQ -> IPC)
L4_PROTO_IRQ_MUX	Protocol for IRQ mux (IRQ -> n x IRQ)
L4_PROTO_SEMAPHORE	Protocol for semaphore objects.
L4_PROTO_META	Meta information protocol.
L4_PROTO_IOMMU	Protocol ID for IO-MMUs.
L4_PROTO_DEBUGGER	Protocol ID for the debugger.
L4_PROTO_SMCCC	Protocol ID for ARM SMCCC calls.

Definition at line 49 of file [types.h](#).

12.84.3.3 L4_platform_ctl_proto

```
enum L4_platform_ctl_proto
```

Predefined protocol type for messages to platform-control objects.

Enumerator

L4_PROTO_PLATFORM_CTL	Protocol messages to a platform control object. See L4_platform_ctl_ops for allowed operations.
-----------------------	---

Definition at line 147 of file [platform_control.h](#).

12.84.4 Function Documentation

12.84.4.1 l4_msgtag()

```
l4_msgtag_t l4_msgtag (
    long label,
    unsigned words,
    unsigned items,
    unsigned flags ) [inline]
```

Create a message tag from the specified values.

Message tag functions.

Parameters

<i>label</i>	The user-defined label
<i>words</i>	The number of untyped words within the UTCB
<i>items</i>	The number of typed items (e.g., flex pages) within the UTCB
<i>flags</i>	The IPC flags for realtime and FPU extensions

Returns

Message tag

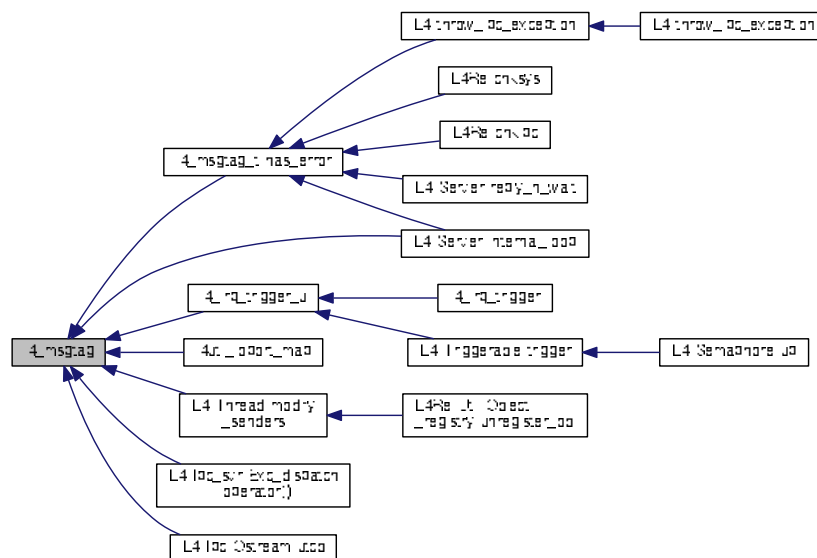
Examples:

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 408 of file [types.h](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), [L4::Server< LOOP_HOOKS >::internal_loop\(\)](#), [l4_irq_trigger_u\(\)](#), [l4util<_ioport_map\(\)](#), [L4::Thread::modify_senders\(\)](#), [L4::lpc_svr::Exc_dispatch< R, Exc >::operator\(\)](#), and [L4::lpc::Ostream::utcb\(\)](#).

Here is the caller graph for this function:



12.84.4.2 l4_msgtag_flags()

```

unsigned l4_msgtag_flags (
    l4_msgtag_t t ) [inline]

```

Get the flags.

The flag are defined by [l4_msgtag_flags](#).

Parameters

<i>t</i>	The tag
----------	---------

Returns

Flags

Definition at line 432 of file [types.h](#).

References [l4_msgtag_t::raw](#).

12.84.4.3 l4_msgtag_has_error()

```
unsigned l4_msgtag_has_error (  
    l4_msgtag_t t ) [inline]
```

Test for error indicator flag.

Parameters

<i>t</i>	The tag
----------	---------

Returns

>0 for yes, 0 for no

Return whether the kernel operation caused a communication error, e.g. with IPC. if true: utcb->error is valid, otherwise utcb->error is not valid

Examples:

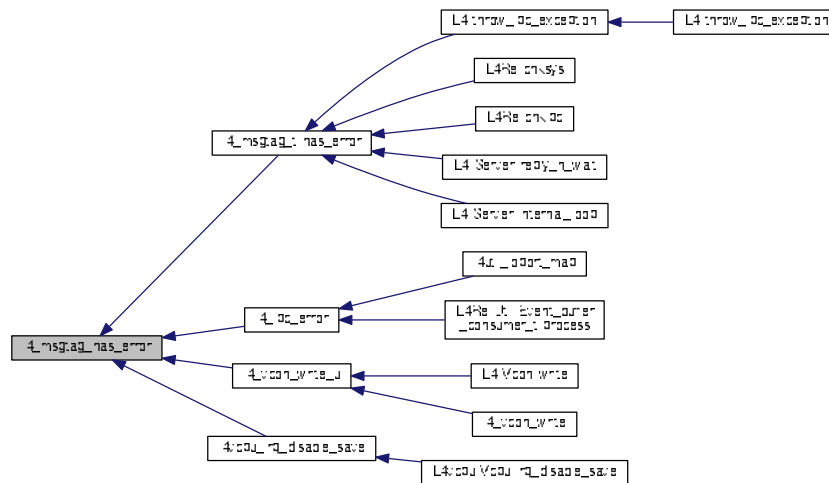
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 437 of file [types.h](#).

References [L4_MSGTAG_ERROR](#), and [l4_msgtag_t::raw](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), [l4_ipc_error\(\)](#), [l4_vcon_write_u\(\)](#), and [l4vcpu_irq_disable_save\(\)](#).

Here is the caller graph for this function:



12.84.4.4 l4_msgtag_is_exception()

```
unsigned l4_msgtag_is_exception (
    l4_msgtag_t t ) [inline]
```

Test for exception protocol.

Parameters

t	The tag
-----	---------

Returns

Boolean value

Examples:

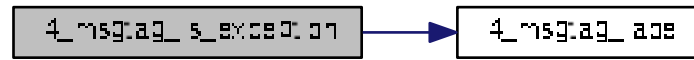
examples/sys/aliens/main.c, examples/sys/singlestep/main.c, and examples/sys/start-with-exc/main.c.

Definition at line 451 of file `types.h`.

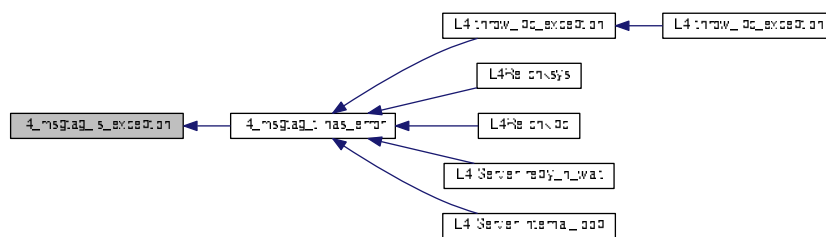
References [l4_msgtag_label\(\)](#), and [L4_PROTO_EXCEPTION](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.84.4.5 l4_msgtag_is_io_page_fault()

```

unsigned l4_msgtag_is_io_page_fault (
    l4_msgtag_t t ) [inline]
  
```

Test for IO-page-fault protocol.

Parameters

<i>t</i>	The tag
----------	---------

Returns

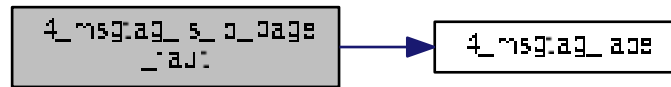
Boolean value

Definition at line 457 of file [types.h](#).

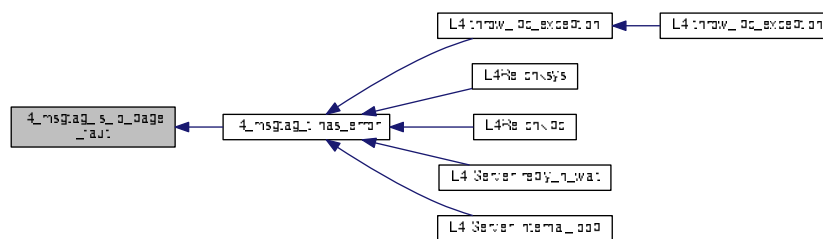
References [l4_msgtag_label\(\)](#), and [L4_PROTO_IO_PAGE_FAULT](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.84.4.6 l4_msgtag_is_page_fault()

```
unsigned l4_msgtag_is_page_fault (
    l4_msgtag_t t ) [inline]
```

Test for page-fault protocol.

Parameters

<code>t</code>	The tag
----------------	---------

Returns

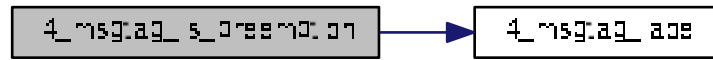
Boolean value

Definition at line 442 of file [types.h](#).

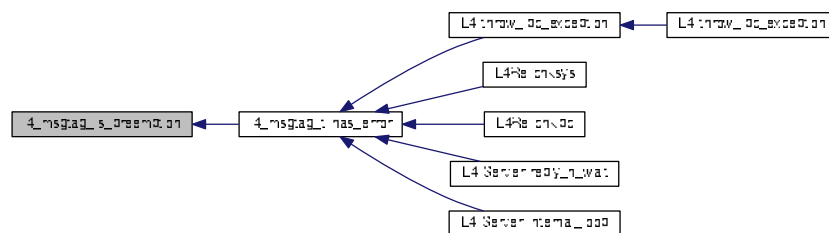
References [l4_msgtag_label\(\)](#), and [L4_PROTO_PAGE_FAULT](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.84.4.8 l4_msgtag_is_sigma0()

```

unsigned l4_msgtag_is_sigma0 (
    l4_msgtag_t t ) [inline]
  
```

Test for sigma0 protocol.

Parameters

<i>t</i>	The tag
----------	---------

Returns

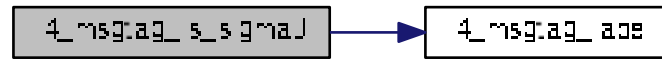
Boolean value

Definition at line 454 of file [types.h](#).

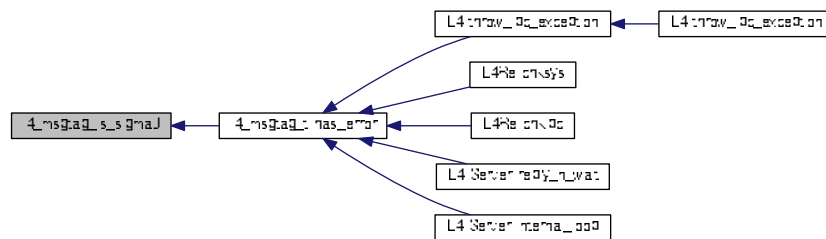
References [l4_msgtag_label\(\)](#), and [L4_PROTO_SIGMA0](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.84.4.9 l4_msgtag_is_sys_exception()

```

unsigned l4_msgtag_is_sys_exception (
    l4_msgtag_t t ) [inline]
  
```

Test for system-exception protocol.

Parameters

<i>t</i>	The tag
----------	---------

Returns

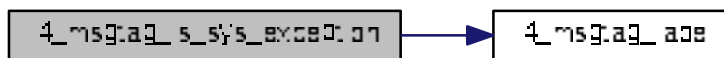
Boolean value

Definition at line 448 of file [types.h](#).

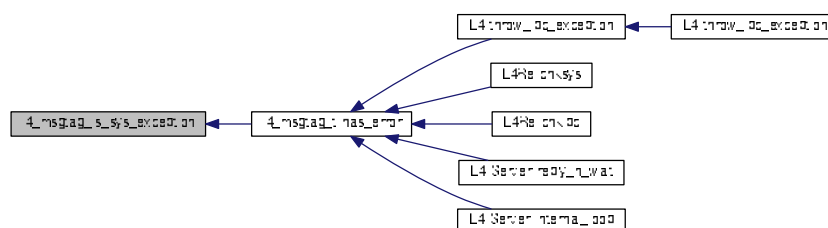
References [l4_msgtag_label\(\)](#), and [L4_PROTO_SYS_EXCEPTION](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.84.4.10 l4_msgtag_items()

```

unsigned l4_msgtag_items (
    l4_msgtag_t t ) [inline]
  
```

Get the number of typed items.

Parameters

<i>t</i>	The tag
----------	---------

Returns

Number of items.

Definition at line 428 of file [types.h](#).

References [l4_msgtag_t::raw](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), and [l4util_ioport_map\(\)](#).

12.84.4.12 l4_msgtag_words()

```
unsigned l4_msgtag_words (  
    l4_msgtag_t t ) [inline]
```

Get the number of untyped words.

Parameters

<i>t</i>	The tag
----------	---------

Returns

Number of words

Examples:

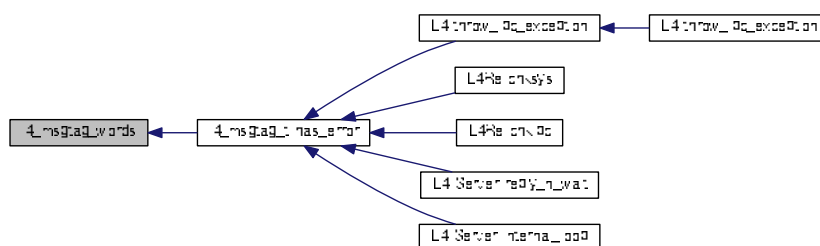
[examples/sys/utcb-ipc/main.c](#).

Definition at line [424](#) of file [types.h](#).

References [l4_msgtag_t::raw](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

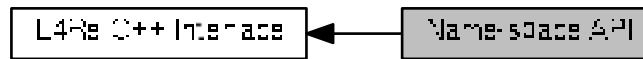
Here is the caller graph for this function:



12.85 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



Data Structures

- class [L4Re::Namespace](#)
Name-space interface.

12.85.1 Detailed Description

API for name spaces that store capabilities.

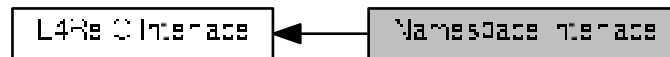
This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

Name spaces play a central role in [L4Re](#), because the implementation of the name space objects determines the policy which capabilities (which objects) are accessible to a client of a name space.

12.86 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:



Enumerations

- enum [l4re_ns_register_flags](#)
Namespace register flags.

Functions

- long [l4re_ns_query_to_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const cap, int timeout) [L4_NOTHROW](#)
Query the name space for the object named by `name`.
- long [l4re_ns_query_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const cap) [L4_NOTHROW](#)
Query the name space for the object named by `name`.
- long [l4re_ns_register_obj_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const obj, unsigned flags) [L4_NOTHROW](#)
Register an object with a name.

12.86.1 Detailed Description

Namespace C interface.

12.86.2 Enumeration Type Documentation

12.86.2.1 l4re_ns_register_flags

```
enum l4re_ns_register_flags
```

Namespace register flags.

See also

[L4Re::Namespace::Register_flags](#)

Definition at line 39 of file [namespace.h](#).

12.86.3 Function Documentation

12.86.3.1 l4re_ns_query_srv()

```
long l4re_ns_query_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const cap ) [inline]
```

Query the name space for the object named by `name`.

Parameters

<i>srv</i>	Name space server to use for the query.
<i>name</i>	String to query.
<i>cap</i>	Capability slot where the received capability will be stored.

Return values

<i>0</i>	Name could be fully resolved.
<i>>0</i>	Name could only be partly resolved. The number of remaining characters is returned.
<i>-L4_ENOENT</i>	Entry could not be found.
<i>-L4_EAGAIN</i>	Entry exists but no object is yet attached. Try again later.
<i><0</i>	IPC errors, see l4_error_code_t .

Definition at line 105 of file [namespace.h](#).

References [EXTERN_C_END](#), and [l4re_ns_query_to_srv\(\)](#).

Here is the call graph for this function:



12.86.3.2 l4re_ns_query_to_srv()

```
long l4re_ns_query_to_srv (
    l4re_namespace_t srv,
```

```
char const * name,  
l4_cap_idx_t const cap,  
int timeout )
```

Query the name space for the object named by `name`.

Parameters

<i>timeout</i>	Timeout of query in milliseconds. The client will only wait if a name already has been registered with the server but no object has been attached yet.
<i>srv</i>	Name space server to use for the query.
<i>name</i>	String to query.
<i>cap</i>	Capability slot where the received capability will be stored.

Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see l4_error_code_t .

Referenced by [l4re_ns_query_srv\(\)](#).

Here is the caller graph for this function:



12.86.3.3 l4re_ns_register_obj_srv()

```

long l4re_ns_register_obj_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const obj,
    unsigned flags )
  
```

Register an object with a name.

Parameters

<i>srv</i>	Name space server to use for the query.
<i>name</i>	Name under which the object should be registered.
<i>obj</i>	Capability to object to register. An invalid capability may be given to only reserve the name for later use.
<i>flags</i>	Flags to assign to the entry, see L4Re::Namespace::Register_flags . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space.

Return values

<i>0</i>	Object was successfully registered with <i>name</i> .
<i>-L4_EEXIST</i>	Name already registered.
<i>-L4_EPERM</i>	Caller doesn't have necessary permissions.
<i>-L4_ENOMEM</i>	Server has insufficient resources.
<i>-L4_EINVAL</i>	Invalid parameter.
<i><0</i>	IPC errors, see l4_error_code_t .

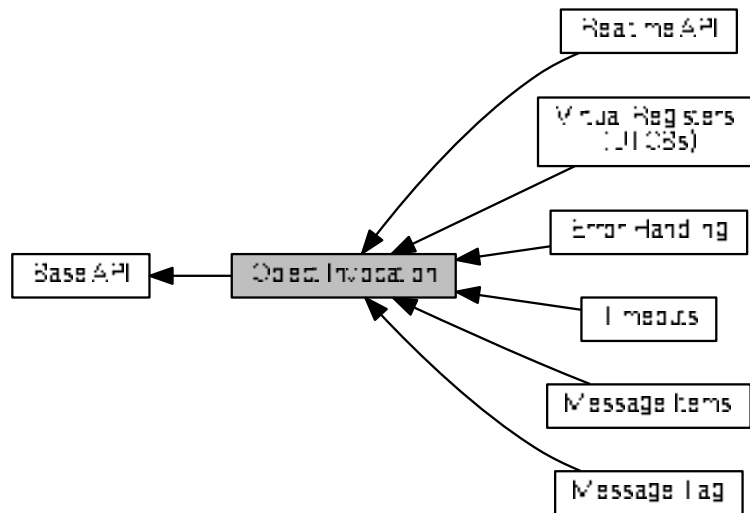
Precondition

requires capability rights: {RW}

12.87 Object Invocation

API for [L4](#) object invocation.

Collaboration diagram for Object Invocation:



Modules

- [Error Handling](#)
Error handling for [L4](#) object invocation.
- [Message Items](#)
Message item related functions.
- [Message Tag](#)
API related to the message tag data type.
- [Realtime API](#)
- [Timeouts](#)
All kinds of timeouts and time related functions.
- [Virtual Registers \(UTCBs\)](#)
[L4](#) Virtual Registers (UTCB).

Files

- file [utcb.h](#)
UTCB definitions.

Enumerations

- enum `l4_syscall_flags_t` {
`L4_SYSF_NONE`, `L4_SYSF_SEND`, `L4_SYSF_RECV`, `L4_SYSF_OPEN_WAIT`,
`L4_SYSF_REPLY`, `L4_SYSF_CALL`, `L4_SYSF_WAIT`, `L4_SYSF_SEND_AND_WAIT`,
`L4_SYSF_REPLY_AND_WAIT` }

Capability selector flags.

Functions

- `l4_msgtag_t l4_ipc_send (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`
*Send a message to an object (do **not** wait for a reply).*
- `l4_msgtag_t l4_ipc_wait (l4_utcb_t *utcb, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`
Wait for an incoming message from any possible sender.
- `l4_msgtag_t l4_ipc_receive (l4_cap_idx_t object, l4_utcb_t *utcb, l4_timeout_t timeout) L4_NOTHROW`
Wait for a message from a specific source.
- `l4_msgtag_t l4_ipc_call (l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`
Object call (usual invocation).
- `l4_msgtag_t l4_ipc_reply_and_wait (l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`
Reply and wait operation (uses the reply capability).
- `l4_msgtag_t l4_ipc_send_and_wait (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`
Send a message and do an open wait.
- `L4_ALWAYS_INLINE l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW`
Generic L4 object invocation.
- `l4_msgtag_t l4_ipc_sleep (l4_timeout_t timeout) L4_NOTHROW`
Sleep for an amount of time.
- `int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTHROW`
Add a flex-page to be sent to the UTCB.

12.87.1 Detailed Description

API for L4 object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

General abstractions for L4 object invocation. The basic principle is that all objects are denoted by a capability that is accessed via a capability selector (see [Capabilities](#)).

This set of functions is common to all kinds of objects provided by the L4 micro kernel. The concrete semantics of an invocation depends on the object that shall be invoked.

Objects may be invoked in various ways, the most common way is to use a *call* operation (`l4_ipc_call()`). However, there are a lot more flavours available that have a semantics depending on the object.

See also

[IPC-Gate API](#)

12.87.2 Enumeration Type Documentation

12.87.2.1 l4_syscall_flags_t

enum `l4_syscall_flags_t`

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

Enumerator

<code>L4_SYSF_NONE</code>	Default flags (call to a kernel object). Using this value as flags in the capability selector for an invocation indicates a call (send and wait for a reply).
<code>L4_SYSF_SEND</code>	Send-phase flag. Setting this flag in a capability selector induces a send phase, this means a message is send to the object denoted by the capability. For receive phase see L4_SYSF_RECV .
<code>L4_SYSF_RECV</code>	Receive-phase flag. Setting this flag in a capability selector induces a receive phase, this means the invoking thread waits for a message from the object denoted by the capability. For a send phase see L4_SYSF_SEND .
<code>L4_SYSF_OPEN_WAIT</code>	Open-wait flag. This flag indicates that the receive operation (see L4_SYSF_RECV) shall be an <i>open wait</i> . <i>Open wait</i> means that the invoking thread shall wait for a message from any possible sender and <i>not</i> from the sender denoted by the capability.
<code>L4_SYSF_REPLY</code>	Reply flag. This flag indicates that the send phase shall use the in-kernel reply capability instead of the capability denoted by the selector index.
<code>L4_SYSF_CALL</code>	Call flags (combines send and receive). Combines L4_SYSF_SEND and L4_SYSF_RECV .
<code>L4_SYSF_WAIT</code>	Wait flags (combines receive and open wait). Combines L4_SYSF_RECV and L4_SYSF_OPEN_WAIT .
<code>L4_SYSF_SEND_AND_WAIT</code>	Send-and-wait flags. Combines L4_SYSF_SEND and L4_SYSF_WAIT .
<code>L4_SYSF_REPLY_AND_WAIT</code>	Reply-and-wait flags. Combines L4_SYSF_SEND , L4_SYSF_REPLY , and L4_SYSF_WAIT .

Definition at line 39 of file [consts.h](#).

12.87.3 Function Documentation

12.87.3.1 l4_ipc()

```
L4_ALWAYS_INLINE l4_msgtag_t l4_ipc (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_umword_t flags,
```

```
14_umword_t slabel,  
14_msgtag_t tag,  
14_umword_t * rlabel,  
14_timeout_t timeout ) [inline]
```

Generic L4 object invocation.

Parameters

	<i>dest</i>	Destination object.
	<i>utcb</i>	UTCB of the caller.
	<i>flags</i>	Invocation flags (see l4_syscall_flags_t).
	<i>slabel</i>	Send label if applicable (may be seen by the receiver).
	<i>tag</i>	Sending message tag.
out	<i>rlabel</i>	Receiving label.
	<i>timeout</i>	Timeout pair (see l4_timeout_t).

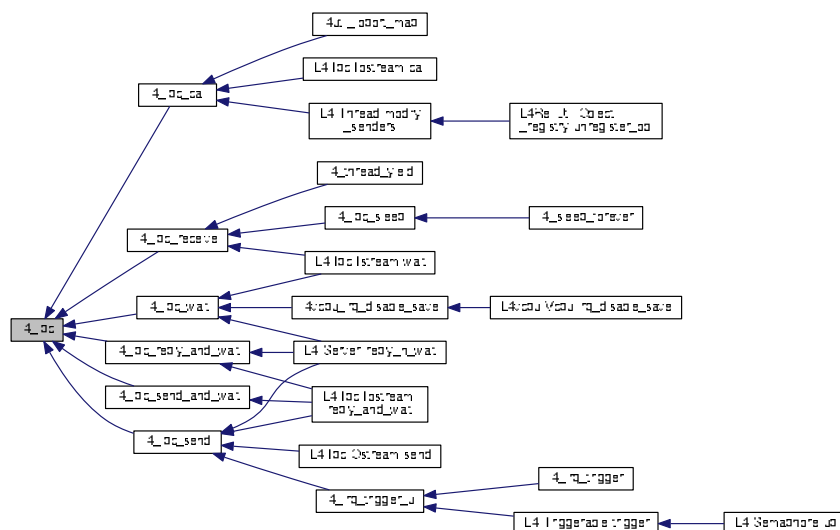
Returns

```
return tag
```

Definition at line 34 of file ipc.h.

Referenced by [l4_ipc_call\(\)](#), [l4_ipc_receive\(\)](#), [l4_ipc_reply_and_wait\(\)](#), [l4_ipc_send\(\)](#), [l4_ipc_send_and_wait\(\)](#), and [l4_ipc_wait\(\)](#).

Here is the caller graph for this function:



12.87.3.2 `l4_ipc_call()`

```
l4_msgtag_t l4_ipc_call (
    l4_cap_idx_t object,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_timeout_t timeout ) [inline]
```

Object call (usual invocation).

Parameters

<i>object</i>	Capability selector for the object to call.
<i>utcb</i>	UTCB of the caller.
<i>tag</i>	Message tag to describe the message to be sent.
<i>timeout</i>	Timeout pair for send an receive phase (see l4_timeout_t).

Returns

result tag

A message is sent to the object and the invoker waits for a reply from the object. Messages from other sources are not accepted.

Note

The send-to-receive transition needs no time, the object can reply with a send timeout of zero.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [445](#) of file [ipc.h](#).

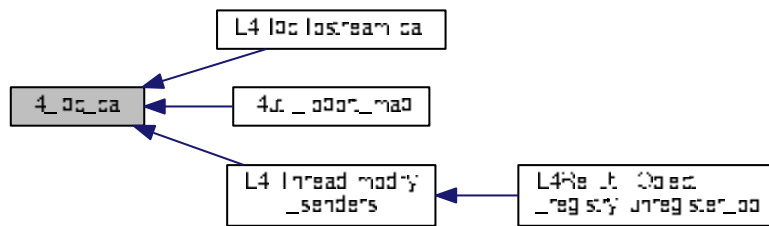
References [l4_ipc\(\)](#), and [L4_SYSF_CALL](#).

Referenced by [L4::ipc::loststream::call\(\)](#), [l4util_ioport_map\(\)](#), and [L4::Thread::modify_senders\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.3.3 l4_ipc_receive()

```

l4_msgtag_t l4_ipc_receive (
    l4_cap_idx_t object,
    l4_utcb_t * utcb,
    l4_timeout_t timeout ) [inline]
  
```

Wait for a message from a specific source.

Parameters

<i>object</i>	Object to receive a message from.
<i>timeout</i>	Timeout pair (see l4_timeout_t , only the receive part matters).
<i>utcb</i>	UTCB of the caller.

Returns

result tag.

This operation waits for a message from the specified object. Messages from other sources are not accepted by this operation. The operation does not include a send phase, this means no message is sent to the object.

Note

This operation is usually used to receive messages from a specific IRQ or thread. However, it is not common to use this operation for normal applications.

Examples:

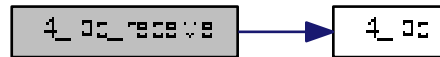
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 487 of file [ipc.h](#).

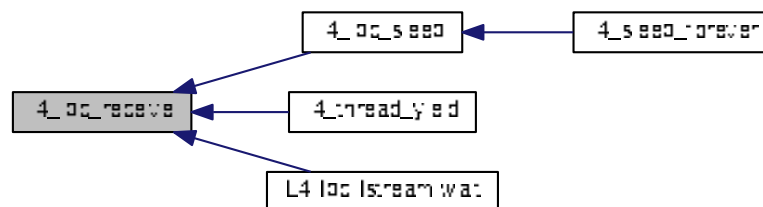
References [l4_ipc\(\)](#), [L4_SYSF_RECV](#), and [l4_msgtag_t::raw](#).

Referenced by [l4_ipc_sleep\(\)](#), [l4_thread_yield\(\)](#), and [L4::lpc::lstream::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.3.4 l4_ipc_reply_and_wait()

```

l4_msgtag_t l4_ipc_reply_and_wait (
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_umword_t * label,
    l4_timeout_t timeout ) [inline]
  
```

Reply and wait operation (uses the *reply* capability).

Parameters

	<i>tag</i>	Describes the message to be sent as reply.
	<i>utcb</i>	UTCB of the caller.
out	<i>label</i>	Label assigned to the source object of the received message.
	<i>timeout</i>	Timeout pair (see l4_timeout_t).

Returns

result tag

A message is sent to the previous caller using the implicit reply capability. Afterwards the invoking thread waits for a message from any source.

Note

This is the standard server operation: it sends a reply to the actual client and waits for the next incoming request, which may come from any other client.

Examples:

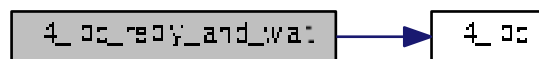
[examples/sys/ipc/ipc_example.c](#).

Definition at line 453 of file [ipc.h](#).

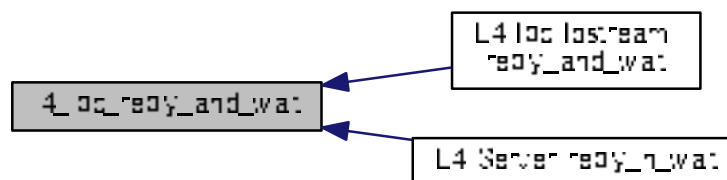
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), and [L4_SYSF_REPLY_AND_WAIT](#).

Referenced by [L4::lpc::lostream::reply_and_wait\(\)](#), and [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**12.87.3.5 l4_ipc_send()**

```

l4_msgtag_t l4_ipc_send (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_timeout_t timeout ) [inline]
  
```

Send a message to an object (do **not** wait for a reply).

Parameters

<i>dest</i>	Capability selector for the destination object.
<i>utcb</i>	UTCB of the caller.
<i>tag</i>	Descriptor for the message to be sent.
<i>timeout</i>	Timeout pair (see l4_timeout_t) only send part is relevant.

Returns

result tag

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

Attention

This is a special-purpose message transfer, objects usually support only invocation via [l4_ipc_call\(\)](#).

Examples:

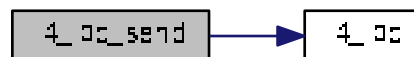
[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 470 of file [ipc.h](#).

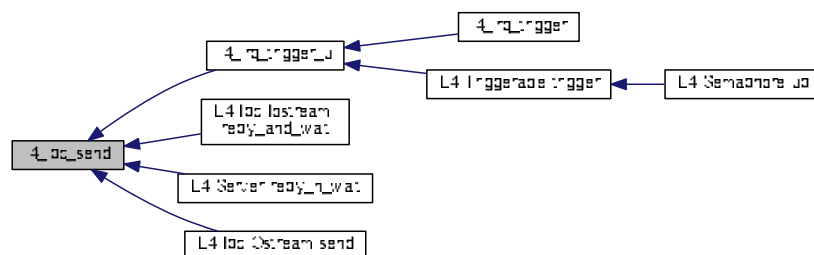
References [l4_ipc\(\)](#), and [L4_SYSF_SEND](#).

Referenced by [l4_irq_trigger_u\(\)](#), [L4::lpc::lstream::reply_and_wait\(\)](#), [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#), and [L4::lpc::Ostream::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.3.6 l4_ipc_send_and_wait()

```
l4_msgtag_t l4_ipc_send_and_wait (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_umword_t * label,
    l4_timeout_t timeout ) [inline]
```

Send a message and do an open wait.

Parameters

	<i>dest</i>	Object to send a message to.
	<i>utcb</i>	UTCB of the caller.
	<i>tag</i>	Describes the message that shall be sent.
out	<i>label</i>	Label assigned to the source object of the receive phase.
	<i>timeout</i>	Timeout pair (see l4_timeout_t).

Returns

result tag

A message is sent to the destination object and the invoking thread waits for a reply from any source.

Note

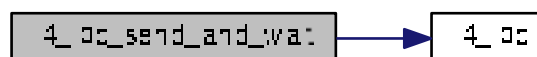
This is a special-purpose operation and shall not be used in general applications.

Definition at line 461 of file [ipc.h](#).

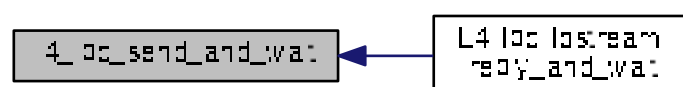
References [l4_ipc\(\)](#), and [L4_SYSF_SEND_AND_WAIT](#).

Referenced by [L4::ipc::lostream::reply_and_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.3.7 l4_ipc_sleep()

```
l4_msgtag_t l4_ipc_sleep (
    l4_timeout_t timeout ) [inline]
```

Sleep for an amount of time.

Parameters

<i>timeout</i>	Timeout pair (see l4_timeout_t , the receive part matters).
----------------	---

Returns

error code:

- [L4_IPC_RETIMEOUT](#): success
- [L4_IPC_RECANCELED](#) woken up by a different thread ([l4_thread_ex_regs\(\)](#)).

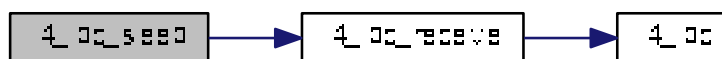
The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4_thread_ex_regs\(\)](#).

Definition at line 496 of file [ipc.h](#).

References [L4_INVALID_CAP](#), and [l4_ipc_receive\(\)](#).

Referenced by [l4_sleep_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.3.8 l4_ipc_wait()

```
l4_msgtag_t l4_ipc_wait (
    l4_utcb_t * utcb,
    l4_umword_t * label,
    l4_timeout_t timeout ) [inline]
```

Wait for an incoming message from any possible sender.

Parameters

<i>utcb</i>	UTCB of the caller.
-------------	---------------------

Return values

<i>label</i>	Label assigned to the source object (IPC gate or IRQ).
--------------	--

Parameters

<i>timeout</i>	Timeout pair (see l4_timeout_t , only the receive part is used).
----------------	--

Returns

return tag

This operation does an open wait, and therefore needs no capability to denote the possible source of a message. This means the calling thread waits for an incoming message from any possible source. There is no send phase included in this operation.

The usual usage of this function is to call that function when entering a server loop in a user-level server that implements user-level objects, see also [l4_ipc_reply_and_wait\(\)](#).

Examples:

[examples/sys/ipc/ipc_example.c](#).

Definition at line [478](#) of file [ipc.h](#).

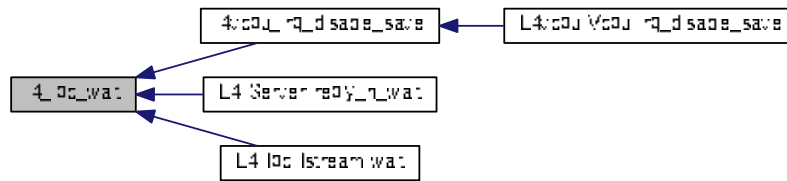
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), [L4_SYSF_WAIT](#), and [l4_msgtag_t::raw](#).

Referenced by [l4vcpu_irq_disable_save\(\)](#), [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#), and [L4::lpc::lstream< >::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.3.9 l4_sndpage_add()

```

int l4_sndpage_add (
    l4_fpage_t const snd_fpage,
    unsigned long snd_base,
    l4_msgtag_t * tag ) [inline]
  
```

Add a flex-page to be sent to the UTCB.

Parameters

<i>snd_fpage</i>	Flex-page.
<i>snd_base</i>	Send base.
<i>tag</i>	Tag to be modified.

Return values

<i>tag</i>	Modified tag, the number of items will be increased, all other values in the tag will be retained.
------------	--

Returns

0 on success, negative error code otherwise

Definition at line 556 of file [ipc.h](#).

12.88 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:



Data Structures

- class [L4Re::Parent](#)
[Parent](#) interface.

12.88.1 Detailed Description

[Parent](#) interface.

The parent interface provides means for an [L4](#) task to signal changes in its execution state. The main purpose is to signal program termination.

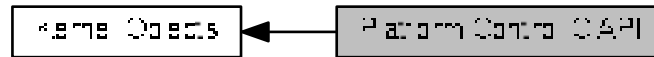
See also

[L4Re::Parent](#) for information about the concrete interface.

12.89 Platform Control C API

C interface for controlling platform-wide properties.

Collaboration diagram for Platform Control C API:



Functions

- [l4_msgtag_t l4_platform_ctl_system_suspend \(l4_cap_idx_t pfc, l4_umword_t extras\) L4_NOTHROW](#)
Enter suspend to RAM.
- [l4_msgtag_t l4_platform_ctl_system_shutdown \(l4_cap_idx_t pfc, l4_umword_t reboot\) L4_NOTHROW](#)
Shutdown or reboot the system.
- [l4_msgtag_t l4_platform_ctl_cpu_enable \(l4_cap_idx_t pfc, l4_umword_t phys_id\) L4_NOTHROW](#)
Enable an offline CPU.
- [l4_msgtag_t l4_platform_ctl_cpu_disable \(l4_cap_idx_t pfc, l4_umword_t phys_id\) L4_NOTHROW](#)
Disable an online CPU.

12.89.1 Detailed Description

C interface for controlling platform-wide properties.

Include File

```
#include <l4/sys/platform_control.h>
```

The API allows a client to suspend, reboot or shutdown the system.

For the C++ interface refer to [L4::Platform_control](#)

12.89.2 Function Documentation

12.89.2.1 l4_platform_ctl_cpu_disable()

```
l4_msgtag_t l4_platform_ctl_cpu_disable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id ) [inline]
```

Disable an online CPU.

Parameters

<i>pfc</i>	Capability to the platform control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.

Returns

System call message tag

Definition at line 232 of file [platform_control.h](#).

12.89.2.2 l4_platform_ctl_cpu_enable()

```
l4_msgtag_t l4_platform_ctl_cpu_enable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id ) [inline]
```

Enable an offline CPU.

Parameters

<i>pfc</i>	Capability to the platform control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.

Returns

System call message tag

Definition at line 225 of file [platform_control.h](#).

12.89.2.3 l4_platform_ctl_system_shutdown()

```
l4_msgtag_t l4_platform_ctl_system_shutdown (
    l4_cap_idx_t pfc,
    l4_umword_t reboot ) [inline]
```

Shutdown or reboot the system.

Parameters

<i>pfc</i>	Capability selector for the platform-control object
<i>reboot</i>	Shutdown when 0, or reboot when 1.

Returns

Syscall return tag

Definition at line 194 of file [platform_control.h](#).

12.89.2.4 l4_platform_ctl_system_suspend()

```
l4_msgtag_t l4_platform_ctl_system_suspend (
    l4_cap_idx_t pfc,
    l4_umword_t extras ) [inline]
```

Enter suspend to RAM.

Parameters

<i>pfc</i>	Capability selector for the platform-control object
<i>extras</i>	some extra platform-specific information needed to enter suspend to RAM.

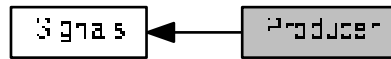
Returns

Syscall return tag

Definition at line 187 of file [platform_control.h](#).

12.90 Producer

Collaboration diagram for Producer:



Functions

- long [l4shmc_trigger](#) (l4shmc_signal_t *signal)
Trigger a signal.

12.90.1 Detailed Description

12.90.2 Function Documentation

12.90.2.1 l4shmc_trigger()

```
long l4shmc_trigger (
    l4shmc_signal_t * signal ) [inline]
```

Trigger a signal.

Parameters

<i>signal</i>	Signal to trigger.
---------------	--------------------

Return values

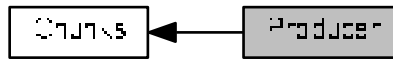
0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.91 Producer

Collaboration diagram for Producer:



Functions

- long [l4shmc_chunk_try_to_take](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy.
- long [l4shmc_chunk_ready](#) (l4shmc_chunk_t *chunk, [l4_umword_t](#) size)
Mark chunk as filled (ready).
- long [l4shmc_chunk_ready_sig](#) (l4shmc_chunk_t *chunk, [l4_umword_t](#) size)
Mark chunk as filled (ready) and signal consumer.
- long [l4shmc_is_chunk_clear](#) (l4shmc_chunk_t *chunk)
Check whether chunk is free.

12.91.1 Detailed Description

12.91.2 Function Documentation

12.91.2.1 l4shmc_chunk_ready()

```

long l4shmc_chunk_ready (
    l4shmc_chunk_t * chunk,
    l4_umword_t size ) [inline]
  
```

Mark chunk as filled (ready).

Parameters

<i>chunk</i>	chunk.
<i>size</i>	Size of data in the chunk, in bytes.

Return values

0	Success.
<0	Error.

12.91.2.2 l4shmc_chunk_ready_sig()

```
long l4shmc_chunk_ready_sig (  
    l4shmc_chunk_t * chunk,  
    l4_umword_t size ) [inline]
```

Mark chunk as filled (ready) and signal consumer.

Parameters

<i>chunk</i>	chunk.
<i>size</i>	Size of data in the chunk, in bytes.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.91.2.3 l4shmc_chunk_try_to_take()

```
long l4shmc_chunk_try_to_take (  
    l4shmc_chunk_t * chunk ) [inline]
```

Try to mark chunk busy.

Parameters

<i>chunk</i>	chunk to mark.
--------------	----------------

Return values

0	Chunk could be taken.
<0	Chunk could not be taken, try again.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.91.2.4 l4shmc_is_chunk_clear()

```
long l4shmc_is_chunk_clear (
    l4shmc_chunk_t * chunk )    [inline]
```

Check whether chunk is free.

Parameters

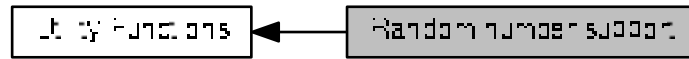
<i>chunk</i>	Chunk to check.
--------------	-----------------

Return values

0	Success.
<0	Error.

12.92 Random number support

Collaboration diagram for Random number support:



Functions

- `l4_uint32_t l4util_rand` (void)
Deliver next random number.
- `void l4util_srand` (`l4_uint32_t` seed)
Initialize random number generator.

12.92.1 Detailed Description

12.92.2 Function Documentation

12.92.2.1 l4util_rand()

```
l4_uint32_t l4util_rand (
    void )
```

Deliver next random number.

Returns

A new random number

12.92.2.2 l4util_srand()

```
void l4util_srand (
    l4_uint32_t seed )
```

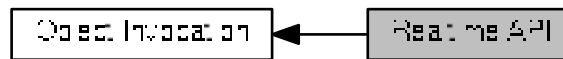
Initialize random number generator.

Parameters

<i>seed</i>	Value to initialize
-------------	---------------------

12.93 Realtime API

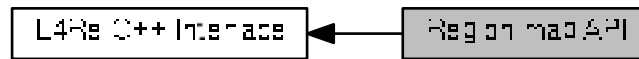
Collaboration diagram for Realtime API:



12.94 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



Data Structures

- class [L4Re::Rm](#)
Region map.

12.94.1 Detailed Description

Virtual address-space management.

The central purpose of the region-map API is to provide means to manage the virtual memory address space of an [L4](#) task. A region-map object implements two protocols. The first protocol is the kernel page-fault protocol, to resolve page faults for threads running in an [L4](#) task. The second protocol is the region-map protocol itself, that allows to attach a data-space object to a region of the virtual address space.

There are two basic concepts provided by a region-map abstraction:

- Regions provide a means to create a view to a data space (or parts of a data space).
- Areas provide a means to reserve areas in a virtual memory address space for special purposes. A reserved area is skipped when searching for an available range of virtual memory, or may be explicitly used to search only within that area.

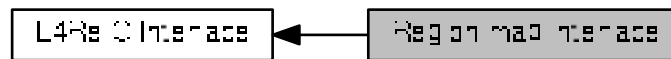
See also

[L4Re::Dataspace](#), [L4Re::Rm](#)

12.95 Region map interface

Region map C interface.

Collaboration diagram for Region map interface:



Enumerations

- enum `l4re_rm_flags_t` {
`L4RE_RM_READ_ONLY` = 0x01, `L4RE_RM_NO_ALIAS` = 0x02, `L4RE_RM_PAGER` = 0x04, `L4RE_RM_RESERVED` = 0x08,
`L4RE_RM_CACHING_SHIFT` = 8, `L4RE_RM_CACHING_DS_SHIFT` = `L4RE_RM_CACHING_SHIFT` - `L4RE_DS_MAP_CACHING_SHIFT`, `L4RE_RM_CACHING` = `L4RE_DS_MAP_CACHING_MASK` << `L4RE_RM_CACHING_DS_SHIFT`, `L4RE_RM_REGION_FLAGS` = `L4RE_RM_CACHING` | 0x0f,
`L4RE_RM_CACHE_NORMAL` = `L4RE_DS_MAP_NORMAL` << `L4RE_RM_CACHING_DS_SHIFT`, `L4RE_RM_CACHE_BUFFERED` = `L4RE_DS_MAP_BUFFERABLE` << `L4RE_RM_CACHING_DS_SHIFT`, `L4RE_RM_CACHE_UNCACHED` = `L4RE_DS_MAP_UNCACHEABLE` << `L4RE_RM_CACHING_DS_SHIFT`,
`L4RE_RM_OVERMAP` = 0x10,
`L4RE_RM_SEARCH_ADDR` = 0x20, `L4RE_RM_IN_AREA` = 0x40, `L4RE_RM_EAGER_MAP` = 0x80, `L4RE_RM_ATTACH_FLAGS` = 0xf0 }

Flags for region operations.

Functions

- int `l4re_rm_reserve_area` (`l4_addr_t` *start, unsigned long size, unsigned flags, unsigned char align) `L4_NOTHROW`
- int `l4re_rm_free_area` (`l4_addr_t` addr) `L4_NOTHROW`
- int `l4re_rm_attach` (void **start, unsigned long size, unsigned long flags, `l4re_ds_t` const mem, `l4_addr_t` offs, unsigned char align) `L4_NOTHROW`
- int `l4re_rm_detach` (void *addr) `L4_NOTHROW`
Detach and unmap in current task.
- int `l4re_rm_detach_ds` (void *addr, `l4re_ds_t` *ds) `L4_NOTHROW`
Detach, unmap and return affected dataspace in current task.
- int `l4re_rm_detach_unmap` (`l4_addr_t` addr, `l4_cap_idx_t` task) `L4_NOTHROW`
Detach and unmap in specified task.
- int `l4re_rm_detach_ds_unmap` (void *addr, `l4re_ds_t` *ds, `l4_cap_idx_t` task) `L4_NOTHROW`
Detach and unmap in specified task.
- int `l4re_rm_find` (`l4_addr_t` *addr, unsigned long *size, `l4_addr_t` *offset, unsigned *flags, `l4re_ds_t` *m) `L4_NOTHROW`
- void `l4re_rm_show_lists` (void) `L4_NOTHROW`
Dump region map internal data structures.

- int [l4re_rm_reserve_area_srv](#) ([l4_cap_idx_t](#) rm, [l4_addr_t](#) *start, unsigned long size, unsigned flags, unsigned char align) [L4_NOTHROW](#)
- int [l4re_rm_free_area_srv](#) ([l4_cap_idx_t](#) rm, [l4_addr_t](#) addr) [L4_NOTHROW](#)
- int [l4re_rm_attach_srv](#) ([l4_cap_idx_t](#) rm, void **start, unsigned long size, unsigned long flags, [l4re_ds_t](#) const mem, [l4_addr_t](#) offs, unsigned char align) [L4_NOTHROW](#)
- int [l4re_rm_detach_srv](#) ([l4_cap_idx_t](#) rm, [l4_addr_t](#) addr, [l4re_ds_t](#) *ds, [l4_cap_idx_t](#) task) [L4_NOTHROW](#)
- int [l4re_rm_find_srv](#) ([l4_cap_idx_t](#) rm, [l4_addr_t](#) *addr, unsigned long *size, [l4_addr_t](#) *offset, unsigned *flags, [l4re_ds_t](#) *m) [L4_NOTHROW](#)
- void [l4re_rm_show_lists_srv](#) ([l4_cap_idx_t](#) rm) [L4_NOTHROW](#)

Dump region map internal data structures.

12.95.1 Detailed Description

Region map C interface.

12.95.2 Enumeration Type Documentation

12.95.2.1 [l4re_rm_flags_t](#)

enum [l4re_rm_flags_t](#)

Flags for region operations.

Enumerator

L4RE_RM_READ_ONLY	Region is read-only.
L4RE_RM_NO_ALIAS	The region contains exclusive memory that is not mapped anywhere else.
L4RE_RM_PAGER	Region has a pager.
L4RE_RM_RESERVED	Region is reserved (blocked)
L4RE_RM_CACHING_SHIFT	Start of region mapper cache bits.
L4RE_RM_CACHING_DS_SHIFT	Shift value for Dataspace to Rm cache bits.
L4RE_RM_CACHING	Mask of all region manager cache bits.
L4RE_RM_REGION_FLAGS	Mask of all region flags.
L4RE_RM_CACHE_NORMAL	Cache bits for normal cacheable memory.
L4RE_RM_CACHE_BUFFERED	Cache bits for buffered (write combining) memory.
L4RE_RM_CACHE_UNCACHED	Cache bits for uncached memory.
L4RE_RM_OVERMAP	Unmap memory already mapped in the region.
L4RE_RM_SEARCH_ADDR	Search for a suitable address range.
L4RE_RM_IN_AREA	Search only in area, or map into area.
L4RE_RM_EAGER_MAP	Eagerly map the attached data space in.
L4RE_RM_ATTACH_FLAGS	Mask of all attach flags.

Definition at line [40](#) of file [rm.h](#).

12.95.3 Function Documentation

12.95.3.1 l4re_rm_attach()

```
int l4re_rm_attach (
    void ** start,
    unsigned long size,
    unsigned long flags,
    l4re_ds_t const mem,
    l4_addr_t offs,
    unsigned char align ) [inline]
```

Parameters

in, out	<i>start</i>	Virtual start address where the region manager shall attach the data space. If L4Re::Rm::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to.
	<i>size</i>	Size of the data space to attach (in bytes)
	<i>flags</i>	Flags, see L4Re::Rm::Attach_flags and L4Re::Rm::Region_flags . If the <code>Eager_map</code> flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails.
	<i>mem</i>	Data space
	<i>offs</i>	Offset into the data space to use
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::Search_addr flag is used.

Return values

0	Success
-L4_ENOENT	No area could be found (see L4Re::Rm::In_area)
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Returns

0 on success, <0 on error

See also

[L4Re::Rm::attach](#)

This function is using the `L4::Env::env()->rm()` service.

Examples:

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 261 of file [rm.h](#).

References [l4re_rm_attach_srv\(\)](#).

Here is the call graph for this function:

**12.95.3.2 l4re_rm_attach_srv()**

```

int l4re_rm_attach_srv (
    l4_cap_idx_t rm,
    void ** start,
    unsigned long size,
    unsigned long flags,
    l4re_ds_t const mem,
    l4_addr_t offs,
    unsigned char align )
  
```

See also

[L4Re::Rm::attach](#)

Referenced by [l4re_rm_attach\(\)](#).

Here is the caller graph for this function:



12.95.3.3 l4re_rm_detach()

```
int l4re_rm_detach (
    void * addr ) [inline]
```

Detach and unmap in current task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 271 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



12.95.3.4 l4re_rm_detach_ds()

```
int l4re_rm_detach_ds (
    void * addr,
    l4re_ds_t * ds ) [inline]
```

Detach, unmap and return affected dataspace in current task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

Return values

<i>ds</i>	Returns dataspace that is affected.
-----------	-------------------------------------

Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Examples:

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 284 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



12.95.3.5 l4re_rm_detach_ds_unmap()

```

int l4re_rm_detach_ds_unmap (
    void * addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task ) [inline]
  
```

Detach and unmap in specified task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

Return values

<i>ds</i>	Returns dataspace that is affected.
-----------	-------------------------------------

Parameters

<i>task</i>	Task to unmap pages from, specify L4_INVALID_CAP to not unmap
-------------	---

Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the L4::Env::env()->rm() service.

Definition at line 291 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



12.95.3.6 l4re_rm_detach_srv()

```

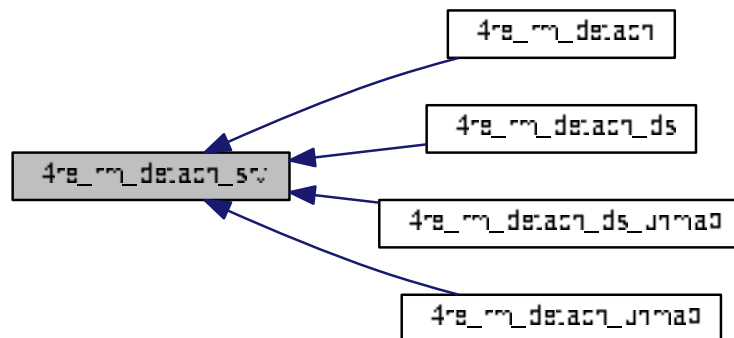
int l4re_rm_detach_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task )
  
```

See also

[L4Re::Rm::detach](#)

Referenced by [l4re_rm_detach\(\)](#), [l4re_rm_detach_ds\(\)](#), [l4re_rm_detach_ds_unmap\(\)](#), and [l4re_rm_detach_unmap\(\)](#).

Here is the caller graph for this function:



12.95.3.7 l4re_rm_detach_unmap()

```

int l4re_rm_detach_unmap (
    l4_addr_t addr,
    l4_cap_idx_t task ) [inline]
  
```

Detach and unmap in specified task.

Parameters

<i>addr</i>	Address of the region to detach.
<i>task</i>	Task to unmap pages from, specify L4_INVALID_CAP to not unmap

Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 278 of file `rm.h`.

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



12.95.3.8 l4re_rm_find()

```

int l4re_rm_find (
    l4_addr_t * addr,
    unsigned long * size,
    l4_addr_t * offset,
    unsigned * flags,
    l4re_ds_t * m ) [inline]
  
```

Returns

0 on success, <0 on error

See also

[L4Re::Rm::find](#)

Definition at line 298 of file `rm.h`.

References [l4re_rm_find_srv\(\)](#).

Here is the call graph for this function:



12.95.3.9 l4re_rm_find_srv()

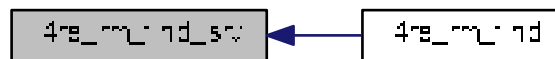
```
int l4re_rm_find_srv (
    l4_cap_idx_t rm,
    l4_addr_t * addr,
    unsigned long * size,
    l4_addr_t * offset,
    unsigned * flags,
    l4re_ds_t * m )
```

See also

[L4Re::Rm::find](#)

Referenced by [l4re_rm_find\(\)](#).

Here is the caller graph for this function:



12.95.3.10 l4re_rm_free_area()

```
int l4re_rm_free_area (
    l4_addr_t addr ) [inline]
```

Returns

0 on success, <0 on error

See also

[L4Re::Rm::free_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 255 of file `rm.h`.

References [l4re_rm_free_area_srv\(\)](#).

Here is the call graph for this function:



12.95.3.11 `l4re_rm_free_area_srv()`

```
int l4re_rm_free_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr )
```

See also

[L4Re::Rm::free_area](#)

Referenced by [l4re_rm_free_area\(\)](#).

Here is the caller graph for this function:

12.95.3.12 `l4re_rm_reserve_area()`

```
int l4re_rm_reserve_area (
    l4_addr_t * start,
    unsigned long size,
    unsigned flags,
    unsigned char align ) [inline]
```

Returns

0 on success, <0 on error

See also

[L4Re::Rm::reserve_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 247 of file [rm.h](#).

References [l4re_rm_reserve_area_srv\(\)](#).

Here is the call graph for this function:



12.95.3.13 l4re_rm_reserve_area_srv()

```
int l4re_rm_reserve_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t * start,
    unsigned long size,
    unsigned flags,
    unsigned char align )
```

See also

[L4Re::Rm::reserve_area](#)

Referenced by [l4re_rm_reserve_area\(\)](#).

Here is the caller graph for this function:



12.95.3.14 l4re_rm_show_lists()

```
void l4re_rm_show_lists (
    void ) [inline]
```

Dump region map internal data structures.

This function is using the `L4::Env::env()->rm()` service.

Definition at line 305 of file [rm.h](#).

References [l4re_rm_show_lists_srv\(\)](#).

Here is the call graph for this function:



12.96 Scheduler

C interface of the Scheduler kernel object.

Collaboration diagram for Scheduler:



Data Structures

- struct [l4_sched_cpu_set_t](#)
CPU sets.
- struct [l4_sched_param_t](#)
Scheduler parameter set.

Typedefs

- typedef struct [l4_sched_cpu_set_t](#) [l4_sched_cpu_set_t](#)
CPU sets.
- typedef struct [l4_sched_param_t](#) [l4_sched_param_t](#)
Scheduler parameter set.

Enumerations

- enum [L4_scheduler_ops](#) { [L4_SCHEDULER_INFO_OP](#) = 0UL, [L4_SCHEDULER_RUN_THREAD_OP](#) = 1↔UL, [L4_SCHEDULER_IDLE_TIME_OP](#) = 2UL }
- Operations on the Scheduler object.*

Functions

- [l4_sched_cpu_set_t](#) [l4_sched_cpu_set](#) ([l4_umword_t](#) offset, unsigned char granularity, [l4_umword_t](#) map=1) [L4_NOTHROW](#)
- [l4_msgtag_t](#) [l4_scheduler_info](#) ([l4_cap_idx_t](#) scheduler, [l4_umword_t](#) *cpu_max, [l4_sched_cpu_set_t](#) *cpus) [L4_NOTHROW](#)
Get scheduler information.
- [l4_sched_param_t](#) [l4_sched_param](#) (unsigned prio, [l4_cpu_time_t](#) quantum=0) [L4_NOTHROW](#)
Construct scheduler parameter.
- [l4_msgtag_t](#) [l4_scheduler_run_thread](#) ([l4_cap_idx_t](#) scheduler, [l4_cap_idx_t](#) thread, [l4_sched_param_t](#) const *sp) [L4_NOTHROW](#)
Run a thread on a Scheduler.
- [l4_msgtag_t](#) [l4_scheduler_idle_time](#) ([l4_cap_idx_t](#) scheduler, [l4_sched_cpu_set_t](#) const *cpus, [l4_kernel_clock_t](#) *us) [L4_NOTHROW](#)
Query the idle time (in μs) of a CPU.
- int [l4_scheduler_is_online](#) ([l4_cap_idx_t](#) scheduler, [l4_umword_t](#) cpu) [L4_NOTHROW](#)
Query if a CPU is online.

12.96.1 Detailed Description

C interface of the Scheduler kernel object.

The Scheduler interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

Include File

```
#include <l4/sys/scheduler.h>
```

12.96.2 Enumeration Type Documentation

12.96.2.1 L4_scheduler_ops

```
enum L4_scheduler_ops
```

Operations on the Scheduler object.

Enumerator

L4_SCHEDULER_INFO_OP	Query infos about the scheduler.
L4_SCHEDULER_RUN_THREAD_OP	Run a thread on this scheduler.
L4_SCHEDULER_IDLE_TIME_OP	Query idle time for the scheduler.

Definition at line 201 of file [scheduler.h](#).

12.96.3 Function Documentation

12.96.3.1 l4_sched_cpu_set()

```
l4_sched_cpu_set_t l4_sched_cpu_set (
    l4_umword_t offset,
    unsigned char granularity,
    l4_umword_t map = 1 ) [inline]
```

Parameters

<i>offset</i>	Offset.
<i>granularity</i>	Granularity in log2 notation.
<i>map</i>	Bitmap of CPUs, defaults to 1 in C++.

Returns

CPU set.

Examples:

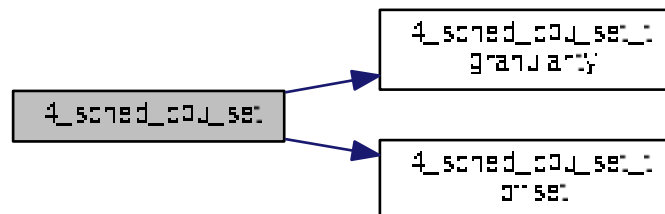
[examples/sys/migrate/thread_migrate.cc](#).

Definition at line 211 of file [scheduler.h](#).

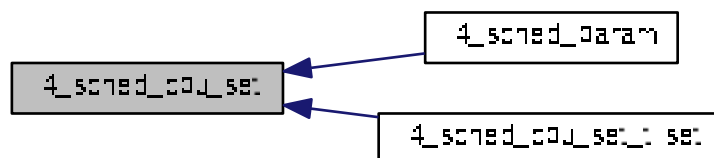
References [l4_sched_cpu_set_t::gran_offset](#), [l4_sched_cpu_set_t::granularity\(\)](#), [l4_sched_cpu_set_t::map](#), and [l4_sched_cpu_set_t::offset\(\)](#).

Referenced by [l4_sched_param\(\)](#), and [l4_sched_cpu_set_t::set\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.96.3.2 l4_scheduler_idle_time()

```

l4_msgtag_t l4_scheduler_idle_time (
    l4_cap_idx_t scheduler,
    l4_sched_cpu_set_t const * cpus,
    l4_kernel_clock_t * us ) [inline]
  
```

Query the idle time (in μ s) of a CPU.

Parameters

	<i>scheduler</i>	Scheduler object.
	<i>cpus</i>	Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried.
out	<i>us</i>	Idle time of queried CPU in μ s.

Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in <code>cpu</code> set.

This function retrieves the idle time in μ s of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate `l` one has to retrieve the idle time at the beginning (`i1`) and the end (`i2`) of a known time interval `t`. The load is then calculated as $l = 1 - (i2 - i1)/t$.

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting -L4_EINVAL or calculating an estimated (incorrect) load of 1.

Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

Definition at line 323 of file [scheduler.h](#).

12.96.3.3 l4_scheduler_info()

```
l4_msgtag_t l4_scheduler_info (
    l4_cap_idx_t scheduler,
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus ) [inline]
```

Get scheduler information.

Parameters

	<i>scheduler</i>	Scheduler object.
out	<i>cpu_max</i>	Maximum number of CPUs ever available.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs.

Return values

0	Success.
-L4_EINVAL	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 309 of file [scheduler.h](#).

12.96.3.4 l4_scheduler_is_online()

```
int l4_scheduler_is_online (
    l4_cap_idx_t scheduler,
    l4_umword_t cpu ) [inline]
```

Query if a CPU is online.

Parameters

<i>scheduler</i>	Scheduler object.
<i>cpu</i>	CPU number whose online status should be queried.

Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 330 of file [scheduler.h](#).

12.96.3.5 l4_scheduler_run_thread()

```
l4_msgtag_t l4_scheduler_run_thread (
    l4_cap_idx_t scheduler,
    l4_cap_idx_t thread,
    l4_sched_param_t const * sp ) [inline]
```

Run a thread on a Scheduler.

Parameters

<i>scheduler</i>	Scheduler object.
<i>thread</i>	Capability of the thread to run.
<i>sp</i>	Scheduling parameters.

Return values

<i>0</i>	Success.
<i>-L4_EINVAL</i>	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

Note

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different physical CPUs.
- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical CPU.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 316 of file [scheduler.h](#).

12.97 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

Namespaces

- [L4::lpc_svr](#)

Helper classes for [L4::Server](#) instantiation.

Data Structures

- class [L4::lpc_svr::Server_iface](#)
Interface for server-loop related functions.
- class [L4::Basic_registry](#)
This registry returns the corresponding server object based on the label of an [lpc_gate](#).
- struct [L4::lpc_svr::Ignore_errors](#)
Mix in for LOOP_HOOKS to ignore IPC errors.
- struct [L4::lpc_svr::Default_timeout](#)
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.
- struct [L4::lpc_svr::Compound_reply](#)
Mix in for LOOP_HOOKS to always use compound reply and wait.
- struct [L4::lpc_svr::Default_setup_wait](#)
Mix in for LOOP_HOOKS for setup_wait no op.
- class [L4::lpc_svr::Timed_work< HOOKS >](#)
DEPRECATED.
- class [L4::lpc_svr::Br_manager_no_buffers](#)
Empty implementation of [Server_iface](#).
- struct [L4::lpc_svr::Default_loop_hooks](#)
Default LOOP_HOOKS.
- class [L4::Server< LOOP_HOOKS >](#)
Basic server loop for handling client requests.
- class [L4::Server_object](#)
Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).
- struct [L4::Server_object_t< IFACE, BASE >](#)
Base class (template) for server implementing server objects.
- struct [L4::Server_object_x< Derived, IFACE, BASE >](#)
Helper class to implement p_dispatch based server objects.
- struct [L4::Irq_handler_object](#)
[Server](#) object base class for handling IRQ messages.
- class [L4::lpc_svr::Timeout](#)
Callback interface for [Timeout_queue](#).
- class [L4::lpc_svr::Timeout_queue](#)
[Timeout](#) queue to be used in l4re server loop.
- class [L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >](#)
Loop hooks mixin for integrating a timeout queue into the server loop.

Enumerations

- enum [L4::lpc_svr::Reply_mode](#) { [L4::lpc_svr::Reply_compound](#), [L4::lpc_svr::Reply_separate](#) }
Reply mode for server loop.

12.97.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

,

12.97.2 Enumeration Type Documentation

12.97.2.1 Reply_mode

enum [L4::Ipc_svr::Reply_mode](#)

Reply mode for server loop.

The reply mode specifies if the server loop shall do a compound reply and wait operation ([Reply_compound](#)), which is the most performant method. Note, `setup_wait()` is called before the reply. The other way is to call reply and wait separately and call `setup_wait` in between.

The actual mode is determined by the return value of the `before_reply()` hook in the `LOOP_HOOKS` of [L4::Server](#).

Enumerator

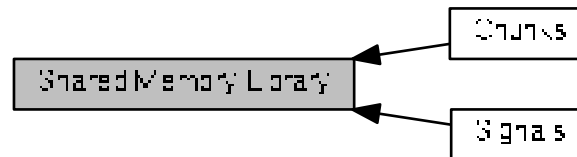
<code>Reply_compound</code>	Server shall use a compound reply and wait (fast).
<code>Reply_separate</code>	Server shall call reply and wait separately.

Definition at line 50 of file [ipc_server_loop](#).

12.98 Shared Memory Library

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

Collaboration diagram for Shared Memory Library:



Modules

- [Chunks](#)
- [Signals](#)

Functions

- long [l4shmc_create](#) (const char *shmc_name, [l4_umword_t](#) shm_size)
Create a shared memory area.
- long [l4shmc_attach](#) (const char *shmc_name, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area.
- long [l4shmc_attach_to](#) (const char *shmc_name, [l4_umword_t](#) timeout_ms, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area, with limited waiting.
- long [l4shmc_connect_chunk_signal](#) ([l4shmc_chunk_t](#) *chunk, [l4shmc_signal_t](#) *signal)
Connect a signal with a chunk.
- long [l4shmc_area_size](#) ([l4shmc_area_t](#) *shmarea)
Get size of shared memory area.
- long [l4shmc_area_size_free](#) ([l4shmc_area_t](#) *shmarea)
Get free size of shared memory area.
- long [l4shmc_area_overhead](#) (void)
Get memory overhead per area that is not available for chunks.
- long [l4shmc_chunk_overhead](#) (void)
Get memory overhead required in addition to the chunk capacity for adding one chunk.

12.98.1 Detailed Description

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

A shared memory area consists of chunks and signals. A chunk is a defined chunk of memory within the memory area with a maximum size. A chunk is filled (written) by a producer and read by a consumer. When a producer has finished writing to the chunk it signals a data ready notification to the consumer.

A consumer attaches to a chunk and waits for the producer to fill the chunk. After reading out the chunk it marks the chunk free again.

A shared memory area can have multiple chunks.

The interface is divided in three roles.

- The master role, responsible for setting up a shared memory area.
- A producer, generating data into a chunk
- A consumer, receiving data.

A signal can be connected with a chunk or can be used independently (e.g. for multiple chunks).

12.98.2 Function Documentation

12.98.2.1 l4shmc_area_overhead()

```
long l4shmc_area_overhead (  
    void )
```

Get memory overhead per area that is not available for chunks.

Returns

size of the overhead in bytes

12.98.2.2 l4shmc_area_size()

```
long l4shmc_area_size (  
    l4shmc_area_t * shmarea ) [inline]
```

Get size of shared memory area.

Parameters

<i>shmarea</i>	Shared memory area.
----------------	---------------------

Return values

>0	Size of the shared memory area.
<0	Error.

12.98.2.3 `l4shmc_area_size_free()`

```
long l4shmc_area_size_free (
    l4shmc_area_t * shmarea )
```

Get free size of shared memory area.

To get the max size to pass to `l4shmc_add_chunk`, subtract `l4shmc_chunk_overhead()`.

Parameters

<i>shmarea</i>	Shared memory area.
----------------	---------------------

Return values

0	Free capacity in the area.
<0	Error.

12.98.2.4 `l4shmc_attach()`

```
long l4shmc_attach (
    const char * shmc_name,
    l4shmc_area_t * shmarea ) [inline]
```

Attach to a shared memory area.

Parameters

	<i>shmc_name</i>	Name of the shared memory area.
out	<i>shmarea</i>	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

Return values

0	Success.
-----	----------

Return values

<0	Error.
------	--------

Examples:

[examples/libs/shmc/prodcons.c](#).

12.98.2.5 l4shmc_attach_to()

```
long l4shmc_attach_to (
    const char * shmc_name,
    l4_umword_t timeout_ms,
    l4shmc_area_t * shmarea )
```

Attach to a shared memory area, with limited waiting.

Parameters

	<i>shmc_name</i>	Name of the shared memory area.
	<i>timeout_ms</i>	Timeout to wait for shm area in milliseconds.
out	<i>shmarea</i>	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

Return values

0	Success.
<0	Error.

12.98.2.6 l4shmc_chunk_overhead()

```
long l4shmc_chunk_overhead (
    void )
```

Get memory overhead required in addition to the chunk capacity for adding one chunk.

Returns

size of the overhead in bytes

12.98.2.7 l4shmc_connect_chunk_signal()

```
long l4shmc_connect_chunk_signal (
    l4shmc_chunk_t * chunk,
    l4shmc_signal_t * signal )
```

Connect a signal with a chunk.

Parameters

<i>chunk</i>	Chunk to attach the signal to.
<i>signal</i>	Signal to attach.

Returns

0 on success, <0 on error

Examples:

[examples/libs/shmc/prodcons.c](#).

12.98.2.8 l4shmc_create()

```
long l4shmc_create (
    const char * shmc_name,
    l4_umword_t shm_size )
```

Create a shared memory area.

Parameters

<i>shmc_name</i>	Name of the shared memory area.
<i>shm_size</i>	Size of the whole shared memory area.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.99 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



Modules

- [Internal constants](#)
Internal sigma0 definitions.

Files

- file [sigma0.h](#)
Sigma0 interface.

Enumerations

- enum [l4sigma0_return_flags_t](#) {
L4SIGMA0_OK, L4SIGMA0_NOTALIGNED, L4SIGMA0_IPCERROR, L4SIGMA0_NOFPAGE ,
L4SIGMA0_SMALLERFPAGE }
Return flags of libsigma0 functions.

Functions

- [l4_kernel_info_t](#) * [l4sigma0_map_kip](#) ([l4_cap_idx_t](#) sigma0, void *addr, unsigned log2_size)
Map the kernel info page from pager to addr.
- int [l4sigma0_map_mem](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) phys, [l4_addr_t](#) virt, [l4_addr_t](#) size)
Request a memory mapping from sigma0.
- int [l4sigma0_map_iomem](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) phys, [l4_addr_t](#) virt, [l4_addr_t](#) size, int cached)
Request IO memory from sigma0.
- int [l4sigma0_map_anypage](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) map_area, unsigned log2_map_size, [l4_addr_t](#) *base, unsigned sz)
Request an arbitrary free page of RAM.
- int [l4sigma0_map_tbuf](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) virt)
Request Fiasco trace buffer.
- void [l4sigma0_debug_dump](#) ([l4_cap_idx_t](#) sigma0)
Request sigma0 to dump internal debug information.
- int [l4sigma0_new_client](#) ([l4_cap_idx_t](#) sigma0, [l4_cap_idx_t](#) gate)
Create a new IPC gate for a new Sigma0 client.
- char const * [l4sigma0_map_errstr](#) (int err)
Get a user readable error messages for the return codes.

12.99.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

12.99.2 Enumeration Type Documentation

12.99.2.1 l4sigma0_return_flags_t

```
enum l4sigma0_return_flags_t
```

Return flags of libsigma0 functions.

Enumerator

L4SIGMA0_OK	Ok.
L4SIGMA0_NOTALIGNED	Phys, virt or size not aligned.
L4SIGMA0_IPCERROR	IPC error.
L4SIGMA0_NOFPAGE	No fpage received.
L4SIGMA0_SMALLERFPAGE	Superpage requested but smaller flexpage received.

Definition at line 81 of file [sigma0.h](#).

12.99.3 Function Documentation

12.99.3.1 l4sigma0_debug_dump()

```
void l4sigma0_debug_dump (  
    l4_cap_idx_t sigma0 )
```

Request sigma0 to dump internal debug information.

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

Parameters

<i>sigma0</i>	the sigma0 thread id.
---------------	-----------------------

12.99.3.2 l4sigma0_map_anypage()

```
int l4sigma0_map_anypage (
    l4_cap_idx_t sigma0,
    l4_addr_t map_area,
    unsigned log2_map_size,
    l4_addr_t * base,
    unsigned sz )
```

Request an arbitrary free page of RAM.

This function requests arbitrary free memory from sigma0. It should be used whenever spare memory is needed, instead of requesting specific physical memory with [l4sigma0_map_mem\(\)](#).

Parameters

<i>sigma0</i>	usually the thread id of sigma0.
<i>map_area</i>	the base address of the local virtual memory area where the page should be mapped.
<i>log2_map_size</i>	the size of the requested page log 2 (the size in bytes is $2^{\text{log2_map_size}}$). This must be at least the minimal page size. By specifying larger sizes the largest possible hardware page size will be used.

Return values

<i>base</i>	physical address of the page received (i.e., the send base of the received mapping if any).
-------------	---

Parameters

<i>sz</i>	Size to map by the server, in 2^{sz} bytes.
-----------	--

Returns

0 on success, !=0 else (see [l4sigma0_map_errstr\(\)](#)).

12.99.3.3 l4sigma0_map_errstr()

```
char const * l4sigma0_map_errstr (
    int err ) [inline]
```

Get a user readable error messages for the return codes.

Parameters

<i>err</i>	the error code reported by the <i>map</i> functions.
------------	--

Returns

a string containing the error message.

Definition at line 213 of file [sigma0.h](#).

References [EXTERN_C_END](#).

12.99.3.4 l4sigma0_map_iomem()

```
int l4sigma0_map_iomem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size,
    int cached )
```

Request IO memory from sigma0.

Parameters

<i>sigma0</i>	Capability to pager implementing the Sigma0 protocol.
<i>phys</i>	The physical address to be requested (page aligned).
<i>virt</i>	The virtual address where the memory should be mapped to (page aligned).
<i>size</i>	The size of the IO memory area to be mapped (multiple of page size)
<i>cached</i>	Requests cacheable IO memory if 1, and uncached if 0.

Return values

<i>0</i>	Success.
<i>-L4SIGMA0_NOTALIGNED</i>	<i>phys</i> , <i>virt</i> , or <i>size</i> are not aligned.
<i>-L4SIGMA0_IPCERROR</i>	IPC error.
<i>-L4SIGMA0_NOFPAGE</i>	No fpage received.

This function is similar to [l4sigma0_map_mem\(\)](#), the difference is that it requests IO memory. IO memory is everything that is not known to be normal RAM. Also ACPI tables or the BIOS memory is treated as IO memory.

See [l4sigma0_map_errstr\(\)](#) to get a description of the return value.

12.99.3.5 l4sigma0_map_kip()

```
l4_kernel_info_t* l4sigma0_map_kip (
    l4_cap_idx_t sigma0,
    void * addr,
    unsigned log2_size )
```

Map the kernel info page from pager to *addr*.

Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>addr</i>	Start of the receive window to receive KIP in.
<i>log2_size</i>	Size of the receive window to receive KIP in.

Returns

Address KIP was mapped to, 0 indicates an error.

12.99.3.6 `l4sigma0_map_mem()`

```
int l4sigma0_map_mem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size )
```

Request a memory mapping from sigma0.

Parameters

<i>sigma0</i>	ID of service talking the sigma0 protocol.
<i>phys</i>	the physical address of the requested page (must be at least aligned to the minimum page size).
<i>virt</i>	the virtual address where the paged should be mapped in the local address space (must be at least aligned to the minimum page size).
<i>size</i>	the size of the requested page, this must be a multiple of the minimum page size.

Returns

0 on success, !=0 else (see [l4sigma0_map_errstr\(\)](#)).

12.99.3.7 `l4sigma0_map_tbuf()`

```
int l4sigma0_map_tbuf (
    l4_cap_idx_t sigma0,
    l4_addr_t virt )
```

Request Fiasco trace buffer.

This is a Fiasco specific feature. Where you can request the kernel internal trace buffer for user-level evaluation. This is for special debugging tools, such as Ferret.

Parameters

<i>sigma0</i>	as usual the sigma0 thread id.
<i>virt</i>	the virtual address where the trace buffer should be mapped,

Returns

0 on success, !=0 else (see [l4sigma0_map_errstr\(\)](#)).

12.99.3.8 l4sigma0_new_client()

```
int l4sigma0_new_client (
    l4_cap_idx_t sigma0,
    l4_cap_idx_t gate )
```

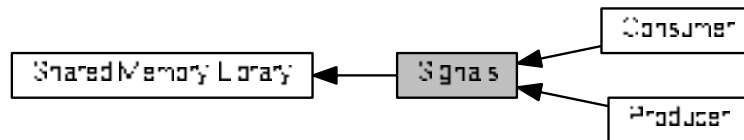
Create a new IPC gate for a new Sigma0 client.

Parameters

<i>sigma0</i>	Capability selector for sigma0 gate.
<i>gate</i>	Capability selector to use for the new gate.

12.100 Signals

Collaboration diagram for Signals:



Modules

- [Consumer](#)
- [Producer](#)

Functions

- long [l4shmc_add_signal](#) (l4shmc_area_t *shmarea, const char *signal_name, l4shmc_signal_t *signal)
Add a signal for the shared memory area.
- long [l4shmc_attach_signal](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, l4shmc_signal_t *signal)
Attach to signal.
- long [l4shmc_attach_signal_to](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, [l4_umword_t](#) timeout_ms, l4shmc_signal_t *signal)
Attach to signal, with timeout.
- long [l4shmc_get_signal_to](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_umword_t](#) timeout_ms, l4shmc_signal_t *signal)
Get signal object from the shared memory area.
- [l4_cap_idx_t](#) [l4shmc_signal_cap](#) (l4shmc_signal_t *signal)
Get the signal capability of a signal.
- long [l4shmc_check_magic](#) (l4shmc_chunk_t *chunk)
Check magic value of a chunk.

12.100.1 Detailed Description

12.100.2 Function Documentation

12.100.2.1 l4shmc_add_signal()

```

long l4shmc_add_signal (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4shmc_signal_t * signal )

```

Add a signal for the shared memory area.

Parameters

	<i>shmarea</i>	The shared memory area to put the chunk in.
	<i>signal_name</i>	Name of the signal.
out	<i>signal</i>	Signal structure to fill in.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.100.2.2 l4shmc_attach_signal()

```
long l4shmc_attach_signal (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4_cap_idx_t thread,
    l4shmc_signal_t * signal ) [inline]
```

Attach to signal.

Parameters

	<i>shmarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
	<i>thread</i>	Thread capability index to attach the signal to.
out	<i>signal</i>	Signal data structure to fill.

Return values

0	Success.
<0	Error.

12.100.2.3 l4shmc_attach_signal_to()

```
long l4shmc_attach_signal_to (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4_cap_idx_t thread,
```

```

14_umword_t timeout_ms,
14shmc_signal_t * signal )

```

Attach to signal, with timeout.

Parameters

	<i>shmbarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
	<i>thread</i>	Thread capability index to attach the signal to.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	<i>signal</i>	Signal data structure to fill.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.100.2.4 l4shmc_check_magic()

```

long l4shmc_check_magic (
    14shmc_chunk_t * chunk ) [inline]

```

Check magic value of a chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	Magic value is not valid.
>0	Chunk is ok, the magic value is valid.

12.100.2.5 l4shmc_get_signal_to()

```

long l4shmc_get_signal_to (
    14shmc_area_t * shmbarea,
    const char * signal_name,

```

```

l4_umword_t timeout_ms,
l4shmc_signal_t * signal )

```

Get signal object from the shared memory area.

Parameters

	<i>shmarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for signal of a chunk to appear in the shared memory area.
out	<i>signal</i>	Signal data structure to fill.

Return values

0	Success.
<0	Error.

12.100.2.6 l4shmc_signal_cap()

```

l4_cap_idx_t l4shmc_signal_cap (
    l4shmc_signal_t * signal ) [inline]

```

Get the signal capability of a signal.

Parameters

<i>signal</i>	Signal.
---------------	---------

Returns

Capability of the signal object.

12.101 Small C++ Template Library

Namespaces

- `cxx`

Our C++ library.

Data Structures

- class `L4::Alloc_list`
A simple list-based allocator.
- class `cxx::Auto_ptr< T >`
Smart pointer with automatic deletion.
- class `cxx::Bitmap_base`
Basic bitmap abstraction.
- class `cxx::Bitmap< BITS >`
A static bit map.
- class `cxx::List_item`
Basic list item.
- struct `cxx::Pair< First, Second >`
Pair of two values.
- class `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`
Basic slab allocator.
- class `cxx::Slab< Type, Slab_size, Max_free, Alloc >`
Slab allocator for object of type Type.
- class `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`
Merged slab allocator (allocators for objects of the same size are merged together).
- class `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >`
Merged slab allocator (allocators for objects of the same size are merged together).
- class `cxx::Nothrow`
Helper type to distinguish the `operator new` version that does not throw exceptions.
- class `cxx::New_allocator< _Type >`
Standard allocator based on `operator new ()`.
- class `L4::String`
A null-terminated string container class.

Functions

- `template<typename T1 >`
`T1 cxx::min (T1 a, T1 b)`
Get the minimum of a and b.
- `template<typename T1 >`
`T1 cxx::max (T1 a, T1 b)`
Get the maximum of a and b.
- `void * operator new (size_t, void *mem, cxx::Nothrow const &) throw ()`
Simple placement new operator.
- `void * operator new (size_t, cxx::Nothrow const &) throw ()`
New operator that does not throw exceptions.

12.101.1 Detailed Description

12.101.2 Function Documentation

12.101.2.1 max()

```
template<typename T1 >
T1 cxx::max (
    T1 a,
    T1 b ) [inline]
```

Get the maximum of *a* and *b*.

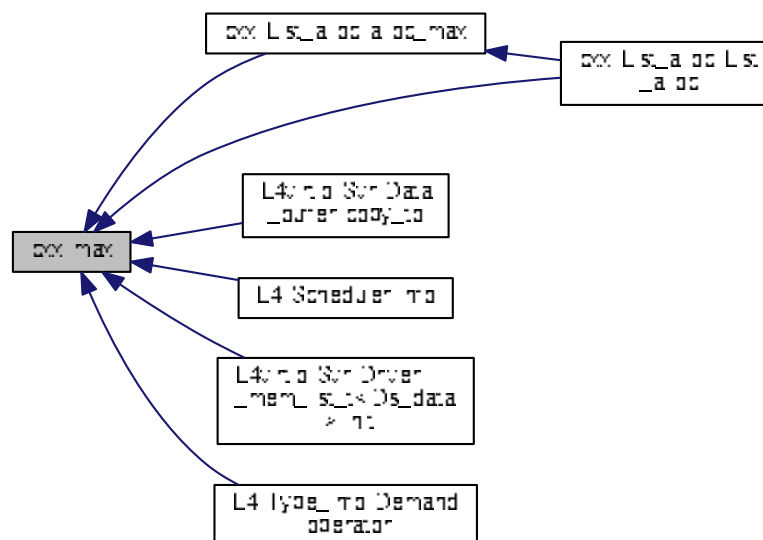
Parameters

<i>a</i>	the first value.
<i>b</i>	the second value.

Definition at line 46 of file [minmax](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [L4::Scheduler::info\(\)](#), [L4virtio::Svr::Driver_mem_list_t<Ds_data>::init\(\)](#), [cxx::List_alloc::List_alloc\(\)](#), and [L4::Type_info::Demand::operator|\(\)](#).

Here is the caller graph for this function:



12.101.2.2 min()

```
template<typename T1 >
T1 cxx::min (
    T1 a,
    T1 b ) [inline]
```

Get the minimum of *a* and *b*.

Parameters

<i>a</i>	the first value.
<i>b</i>	the second value.

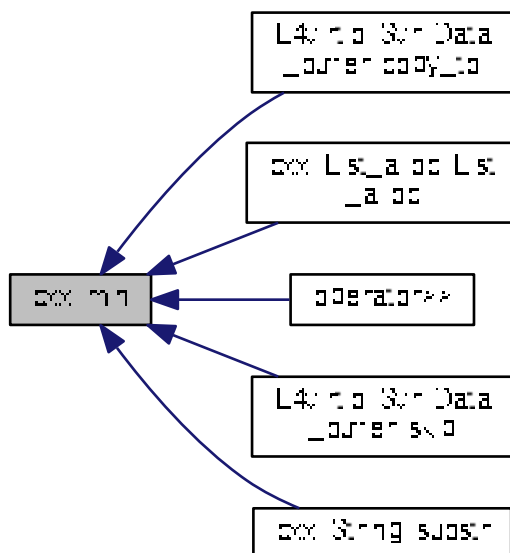
Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 35 of file [minmax](#).

Referenced by [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [cxx::List_alloc::List_alloc\(\)](#), [operator>>\(\)](#), [L4virtio::Svr::Data_buffer::skip\(\)](#), and [cxx::String::substr\(\)](#).

Here is the caller graph for this function:



12.101.2.3 operator new()

```
void* operator new (
    size_t ,
    void * mem,
    cxx::Nothrow const & ) throw )    [inline]
```

Simple placement new operator.

Parameters

<i>mem</i>	the address of the memory block to place the new object.
------------	--

Returns

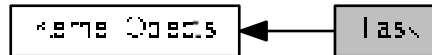
the address given by *mem*.

Definition at line 39 of file [std_alloc](#).

12.102 Task

C interface of the Task kernel object.

Collaboration diagram for Task:



Enumerations

- enum `l4_unmap_flags_t` { `L4_FP_ALL_SPACES`, `L4_FP_DELETE_OBJ`, `L4_FP_OTHER_SPACES` }
Flags for the unmap operation.

Functions

- `l4_msgtag_t l4_task_map (l4_cap_idx_t dst_task, l4_cap_idx_t src_task, l4_fpage_t snd_fpage, l4_addr_t snd_base) L4_NOTHROW`
Map resources available in the source task to a destination task.
- `l4_msgtag_t l4_task_unmap (l4_cap_idx_t task, l4_fpage_t fpage, l4_umword_t map_mask) L4_NOTHROW`
Revoke rights from the task.
- `l4_msgtag_t l4_task_unmap_batch (l4_cap_idx_t task, l4_fpage_t const *fpages, unsigned num_fpages, unsigned long map_mask) L4_NOTHROW`
Revoke rights from a task.
- `l4_msgtag_t l4_task_delete_obj (l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW`
Release capability and delete object.
- `l4_msgtag_t l4_task_release_cap (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`
Release capability.
- `l4_msgtag_t l4_task_cap_valid (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`
Check whether a capability is present (refers to an object).
- `l4_msgtag_t l4_task_cap_has_child (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`
Test whether a capability has child mappings (in another task).
- `l4_msgtag_t l4_task_cap_equal (l4_cap_idx_t task, l4_cap_idx_t cap_a, l4_cap_idx_t cap_b) L4_NOTHROW`
Test whether two capabilities point to the same object with the same rights.
- `l4_msgtag_t l4_task_add_ku_mem (l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW`
Add kernel-user memory.

12.102.1 Detailed Description

C interface of the Task kernel object.

A task represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space.

Task objects are created using the [Factory](#) interface.

Include File

```
#include <l4/sys/task.h>
```

12.102.2 Enumeration Type Documentation

12.102.2.1 l4_unmap_flags_t

```
enum l4_unmap_flags_t
```

Flags for the unmap operation.

See also

[L4::Task::unmap\(\)](#) and [l4_task_unmap\(\)](#)

Enumerator

L4_FP_ALL_SPACES	<p>Flag to tell the unmap operation to unmap all child mappings including the mapping in the invoked task.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p>
L4_FP_DELETE_OBJ	<p>Flag that indicates that the unmap operation on a capability shall try to delete the corresponding objects immediately.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p>
L4_FP_OTHER_SPACES	<p>Counterpart to L4_FP_ALL_SPACES, unmap only child mappings.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p>

Definition at line 157 of file [consts.h](#).

12.102.3 Function Documentation

12.102.3.1 l4_task_add_ku_mem()

```
l4_msgtag_t l4_task_add_ku_mem (
    l4_cap_idx_t task,
    l4_fpage_t ku_mem ) [inline]
```

Add kernel-user memory.

Parameters

<i>task</i>	Capability selector of the task to add the memory to
<i>ku_mem</i>	Flexpage describing the virtual area the memory goes to.

Returns

Syscall return tag

Definition at line 439 of file [task.h](#).

12.102.3.2 l4_task_cap_equal()

```
l4_msgtag_t l4_task_cap_equal (
    l4_cap_idx_t task,
    l4_cap_idx_t cap_a,
    l4_cap_idx_t cap_b ) [inline]
```

Test whether two capabilities point to the same object with the same rights.

Parameters

<i>task</i>	Capability selector of the destination task to do the lookup in
<i>cap_a</i>	Capability selector to compare
<i>cap_b</i>	Capability selector to compare

Returns

label contains 1 if equal, 0 if not equal

Definition at line 432 of file [task.h](#).

12.102.3.3 l4_task_cap_has_child()

```
l4_msgtag_t l4_task_cap_has_child (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
```

Test whether a capability has child mappings (in another task).

Parameters

<i>task</i>	Capability selector of the destination task to do the lookup in
<i>cap</i>	Capability selector to look up in the destination task

Returns

label contains 1 if it has at least one child, 0 if not or invalid

Deprecated Do not use. Undetermined future, might be removed.

Definition at line 426 of file [task.h](#).

12.102.3.4 l4_task_cap_valid()

```
l4_msgtag_t l4_task_cap_valid (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
```

Check whether a capability is present (refers to an object).

Parameters

<i>task</i>	Task to check the capability in.
<i>cap</i>	Valid capability to check for presence.

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Precondition

cap must be a valid capability index (i.e. not L4_INVALID_CAP or the like).

Definition at line 420 of file [task.h](#).

12.102.3.5 `l4_task_delete_obj()`

```
l4_msgtag_t l4_task_delete_obj (
    l4_cap_idx_t task,
    l4_cap_idx_t obj ) [inline]
```

Release capability and delete object.

Parameters

<i>task</i>	Capability selector of destination task
<i>obj</i>	Capability selector of object to delete

Returns

Syscall return tag

The object will be deleted if the obj has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

This operation calls `l4_task_unmap()` with `L4_FP_DELETE_OBJ`.

Definition at line 399 of file `task.h`.

12.102.3.6 `l4_task_map()`

```
l4_msgtag_t l4_task_map (
    l4_cap_idx_t dst_task,
    l4_cap_idx_t src_task,
    l4_fpage_t snd_fpage,
    l4_addr_t snd_base ) [inline]
```

Map resources available in the source task to a destination task.

Parameters

<i>dst_task</i>	Capability selector of destination task
<i>src_task</i>	Capability selector of source task
<i>snd_fpage</i>	Send flexpage that describes an area in the address space or object space of the source task.
<i>snd_base</i>	Send base that describes an offset in the receive window of the destination task.

Returns

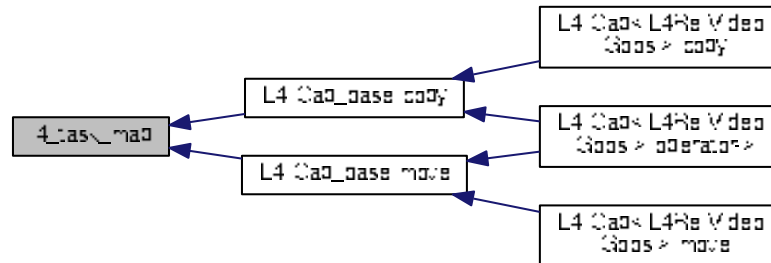
Syscall return tag

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects.

Definition at line 369 of file `task.h`.

Referenced by [L4::Cap_base::copy\(\)](#), and [L4::Cap_base::move\(\)](#).

Here is the caller graph for this function:



12.102.3.7 l4_task_release_cap()

```
l4_msgtag_t l4_task_release_cap (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
```

Release capability.

Parameters

<i>task</i>	Capability selector of destination task
<i>cap</i>	Capability selector to release

Returns

Syscall return tag

This operation unmaps the capability from the specified task.

Definition at line 414 of file [task.h](#).

12.102.3.8 l4_task_unmap()

```
l4_msgtag_t l4_task_unmap (
    l4_cap_idx_t task,
    l4_fpage_t fpage,
    l4_umword_t map_mask ) [inline]
```

Revoke rights from the task.

Parameters

<i>task</i>	Capability selector of destination task
<i>fpage</i>	Flexpage that describes an area in the address space or object space of the destination task
<i>map_mask</i>	Unmap mask, see l4_unmap_flags_t

Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

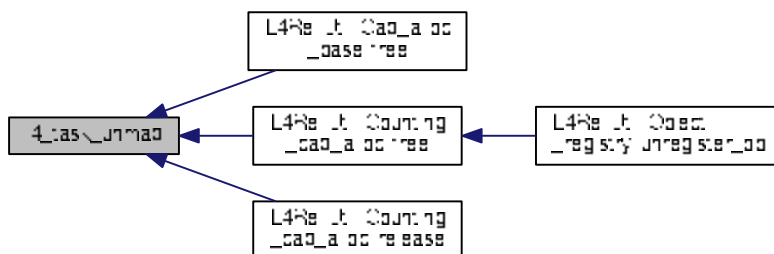
Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line 376 of file [task.h](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free\(\)](#), and [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::release\(\)](#).

Here is the caller graph for this function:



12.102.3.9 l4_task_unmap_batch()

```

l4_msgtag_t l4_task_unmap_batch (
    l4_cap_idx_t task,
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    unsigned long map_mask ) [inline]
  
```

Revoke rights from a task.

Parameters

<i>task</i>	Capability selector of destination task
<i>fpages</i>	An array of flexpages that describes an area in the address space or object space of the destination task each
<i>num_fpages</i>	The size of the fpages array in elements (number of fpages sent).
<i>map_mask</i>	Unmap mask, see l4_unmap_flags_t

Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

The caller needs to take care that num_fpages is not bigger than L4_UTCB_GENERIC_DATA_SIZE - 2.

Note

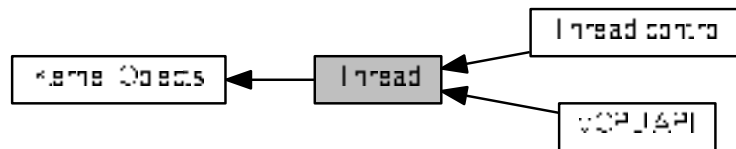
Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line [383](#) of file [task.h](#).

12.103 Thread

Thread object.

Collaboration diagram for Thread:



Modules

- [Thread control](#)
API for Thread Control method.
- [vCPU API](#)
vCPU API

Enumerations

- enum [L4_thread_control_flags](#) {
[L4_THREAD_CONTROL_SET_PAGER](#) = 0x0010000, [L4_THREAD_CONTROL_BIND_TASK](#) = 0x0200000,
[L4_THREAD_CONTROL_ALIEN](#) = 0x0400000, [L4_THREAD_CONTROL_UX_NATIVE](#) = 0x0800000,
[L4_THREAD_CONTROL_SET_EXC_HANDLER](#) = 0x1000000 }
Flags for the thread control operation.
- enum [L4_thread_control_mr_indices](#) {
[L4_THREAD_CONTROL_MR_IDX_FLAGS](#) = 0, [L4_THREAD_CONTROL_MR_IDX_PAGER](#) = 1, [L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER](#) = 2, [L4_THREAD_CONTROL_MR_IDX_FLAG_VALS](#) = 4,
[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB](#) = 5, [L4_THREAD_CONTROL_MR_IDX_BIND_TASK](#) = 6
 }
Indices for the values in the message register for thread control.
- enum [L4_thread_ex_regs_flags](#) { [L4_THREAD_EX_REGS_CANCEL](#) = 0x10000UL, [L4_THREAD_EX_REGS_TRIGGER_EXCEPTION](#) = 0x20000UL }
Flags for the thread ex-regs operation.

Functions

- [l4_msgtag_t l4_thread_ex_regs](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) ip, [l4_addr_t](#) sp, [l4_umword_t](#) flags) [L4_NOTHROW](#)
Exchange basic thread registers.
- [l4_msgtag_t l4_thread_ex_regs_u](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) ip, [l4_addr_t](#) sp, [l4_umword_t](#) flags, [l4_umword_t](#) *utcb) [L4_NOTHROW](#)
Exchange basic thread registers.

- [l4_msgtag_t l4_thread_ex_regs_ret](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) *ip, [l4_addr_t](#) *sp, [l4_umword_t](#) *flags) [L4_NOTHROW](#)
Exchange basic thread registers and return previous values.
- [l4_msgtag_t l4_thread_ex_regs_ret_u](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) *ip, [l4_addr_t](#) *sp, [l4_umword_t](#) *flags, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Exchange basic thread registers and return previous values.
- [l4_msgtag_t l4_thread_yield](#) (void) [L4_NOTHROW](#)
Yield current time slice.
- [l4_msgtag_t l4_thread_switch](#) ([l4_cap_idx_t](#) to_thread) [L4_NOTHROW](#)
Switch to another thread (and donate the remaining time slice).
- [l4_msgtag_t l4_thread_stats_time](#) ([l4_cap_idx_t](#) thread, [l4_kernel_clock_t](#) *us) [L4_NOTHROW](#)
Get consumed time of thread in μ s.
- [l4_msgtag_t l4_thread_vcpu_resume_start](#) (void) [L4_NOTHROW](#)
vCPU return from event handler.
- [l4_msgtag_t l4_thread_vcpu_resume_commit](#) ([l4_cap_idx_t](#) thread, [l4_msgtag_t](#) tag) [L4_NOTHROW](#)
Commit vCPU resume.
- [l4_msgtag_t l4_thread_vcpu_control](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) vcpu_state) [L4_NOTHROW](#)
Enable or disable the vCPU feature for the thread.
- [l4_msgtag_t l4_thread_vcpu_control_u](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) vcpu_state, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Enable or disable the vCPU feature for the thread.
- [l4_msgtag_t l4_thread_vcpu_control_ext](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) ext_vcpu_state) [L4_NOTHROW](#)
Enable or disable the extended vCPU feature for the thread.
- [l4_msgtag_t l4_thread_vcpu_control_ext_u](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) ext_vcpu_state, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Enable or disable the extended vCPU feature for the thread.
- [l4_msgtag_t l4_thread_register_del_irq](#) ([l4_cap_idx_t](#) thread, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Register an IRQ that will trigger upon deletion events.
- [l4_msgtag_t l4_thread_modify_sender_start](#) (void) [L4_NOTHROW](#)
Start a thread sender modification sequence.
- [int l4_thread_modify_sender_add](#) ([l4_umword_t](#) match_mask, [l4_umword_t](#) match, [l4_umword_t](#) del_bits, [l4_umword_t](#) add_bits, [l4_msgtag_t](#) *tag) [L4_NOTHROW](#)
Add a modification pattern to a sender modification sequence.
- [l4_msgtag_t l4_thread_modify_sender_commit](#) ([l4_cap_idx_t](#) thread, [l4_msgtag_t](#) tag) [L4_NOTHROW](#)
Apply (commit) a sender modification sequence.
- [l4_msgtag_t l4_thread_arm_set_tpidruro](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) tpidruro) [L4_NOTHROW](#)
Set the TPIDRURO thread specific register.

12.103.1 Detailed Description

Thread object.

An [L4](#) thread is a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. Thread kernel objects are created using a factory, see [Factory](#) ([l4_factory_create_thread\(\)](#)).

Amongst other things an [L4](#) thread encapsulates:

- CPU state
 - General-purpose registers
 - Program counter

- Stack pointer
- FPU state
- Scheduling parameters, see the [Scheduler](#) API
- Execution state
 - Blocked, Runnable, Running

Thread objects provide an API for

- Thread configuration and manipulation
- Thread switching.

The thread control functions are used to control various aspects of a thread. See [l4_thread_control_start\(\)](#) for more information.

Include File

```
#include <l4/sys/thread.h>
```

For the C++ interface refer to [L4::Thread](#).

12.103.2 Enumeration Type Documentation

12.103.2.1 L4_thread_control_flags

```
enum L4_thread_control_flags
```

Flags for the thread control operation.

Enumerator

L4_THREAD_CONTROL_SET_PAGER	The pager will be given.
L4_THREAD_CONTROL_BIND_TASK	The task to bind the thread to will be given.
L4_THREAD_CONTROL_ALIEN	Alien state of the thread is set.
L4_THREAD_CONTROL_UX_NATIVE	Fiasco-UX only: pass-through of host system calls is set.
L4_THREAD_CONTROL_SET_EXC_HANDLER	The exception handler of the thread will be given.

Definition at line [640](#) of file [thread.h](#).

12.103.2.2 L4_thread_control_mr_indices

```
enum L4_thread_control_mr_indices
```

Indices for the values in the message register for thread control.

Enumerator

L4_THREAD_CONTROL_MR_IDX_FLAGS	See also L4_thread_control_flags .
L4_THREAD_CONTROL_MR_IDX_PAGER	Index for pager cap.
L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER	Index for exception handler.
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS	Index for feature values.
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB	Index for UTCB address for bind.
L4_THREAD_CONTROL_MR_IDX_BIND_TASK	Index for task flex-page for bind.

Definition at line 663 of file [thread.h](#).

12.103.2.3 L4_thread_ex_regs_flags

```
enum L4_thread_ex_regs_flags
```

Flags for the thread ex-regs operation.

Enumerator

L4_THREAD_EX_REGS_CANCEL	Cancel ongoing IPC in the thread.
L4_THREAD_EX_REGS_TRIGGER_EXCEPTION	Trigger artificial exception in thread.

Definition at line 678 of file [thread.h](#).

12.103.3 Function Documentation

12.103.3.1 l4_thread_arm_set_tpidruro()

```
l4_msgtag_t l4_thread_arm_set_tpidruro (
    l4_cap_idx_t thread,
    l4_addr_t tpidruro ) [inline]
```

Set the TPIDRURO thread specific register.

Parameters

<i>thread</i>	Thread to manipulate
<i>tpidruro</i>	The value to be set

Returns

System call return tag

Definition at line 59 of file [thread.h](#).

12.103.3.2 l4_thread_ex_regs()

```
l4_msgtag_t l4_thread_ex_regs (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags ) [inline]
```

Exchange basic thread registers.

Parameters

<i>thread</i>	Capability selector of the thread to manipulate.
<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags .

Returns

System call return tag

This method allows to manipulate a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

The thread is started using [l4_scheduler_run_thread\(\)](#). However, if at the time [l4_scheduler_run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [l4_thread_ex_regs\(\)](#) with a valid instruction pointer might start the thread.

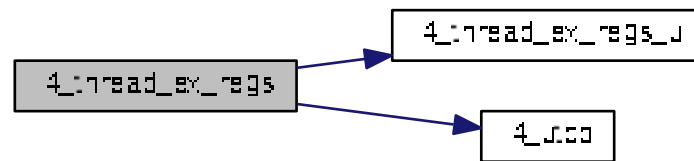
Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 829 of file [thread.h](#).

References [l4_thread_ex_regs_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.103.3.3 l4_thread_ex_regs_ret()

```

l4_msgtag_t l4_thread_ex_regs_ret (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags ) [inline]
  
```

Exchange basic thread registers and return previous values.

Parameters

	<i>thread</i>	Capability selector of the thread to manipulate.
in, out	<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags , return previous CPU flags of the thread.

Returns

System call return tag

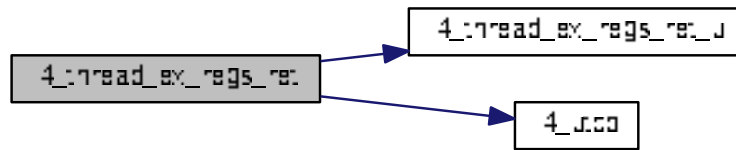
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

Returned values are valid only if function returns successfully.

Definition at line 836 of file [thread.h](#).

References [l4_thread_ex_regs_ret_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.103.3.4 l4_thread_ex_regs_ret_u()

```

l4_msgtag_t l4_thread_ex_regs_ret_u (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags,
    l4_utcb_t * utcb ) [inline]
  
```

Exchange basic thread registers and return previous values.

Parameters

	<i>thread</i>	Capability selector of the thread to manipulate.
in, out	<i>ip</i>	New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags , return previous CPU flags of the thread.
	<i>utcb</i>	UTCB to use for this operation.

Returns

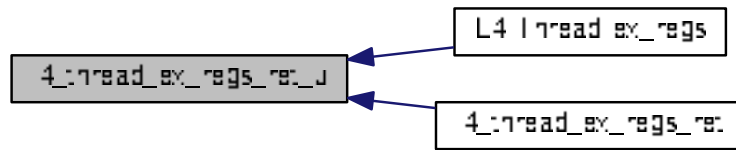
System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

Definition at line 702 of file [thread.h](#).

Referenced by [L4::Thread::ex_regs\(\)](#), and [l4_thread_ex_regs_ret\(\)](#).

Here is the caller graph for this function:



12.103.3.5 l4_thread_ex_regs_u()

```

l4_msgtag_t l4_thread_ex_regs_u (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb ) [inline]
  
```

Exchange basic thread registers.

Parameters

<i>thread</i>	Capability selector of the thread to manipulate.
<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags .
<i>utcb</i>	UTCB to use for this operation.

Returns

System call return tag

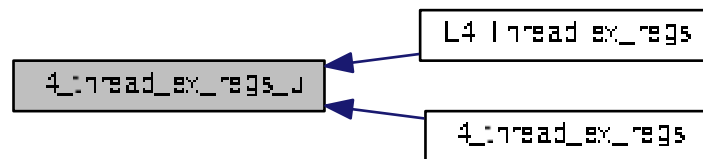
This method allows to manipulate a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

The thread is started using [L4::Scheduler::run_thread\(\)](#). However, if at the time [L4::Scheduler::run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 691 of file [thread.h](#).

Referenced by [L4::Thread::ex_regs\(\)](#), and [l4_thread_ex_regs\(\)](#).

Here is the caller graph for this function:



12.103.3.6 l4_thread_modify_sender_add()

```

int l4_thread_modify_sender_add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits,
    l4_msgtag_t * tag ) [inline]
  
```

Add a modification pattern to a sender modification sequence.

Parameters

<i>tag</i>	Tag received from l4_thread_modify_sender_start() or previous l4_thread_modify_sender_add() calls from the same sequence.
<i>match_mask</i>	Bitmask of bits to match the label.
<i>match</i>	Bitmask that must be equal to the label after applying <i>match_mask</i> .
<i>del_bits</i>	Bits to be deleted from the label.
<i>add_bits</i>	Bits to be added to the label.

Returns

0 on success, <0 on error

In pseudo code: if ((sender_label & match_mask) == match) { label = (label & ~del_bits) | add_bits; }

Only the first match is applied.

See also

[l4_thread_modify_sender_start](#)
[l4_thread_modify_sender_commit](#)

Definition at line 1009 of file [thread.h](#).

12.103.3.7 `l4_thread_modify_sender_commit()`

```
l4_msgtag_t l4_thread_modify_sender_commit (
    l4_cap_idx_t thread,
    l4_msgtag_t tag ) [inline]
```

Apply (commit) a sender modification sequence.

See also

[l4_thread_modify_sender_start](#)
[l4_thread_modify_sender_add](#)

Definition at line 1020 of file [thread.h](#).

12.103.3.8 `l4_thread_modify_sender_start()`

```
l4_msgtag_t l4_thread_modify_sender_start (
    void ) [inline]
```

Start a thread sender modification sequence.

Add modification rules with [l4_thread_modify_sender_add\(\)](#) and commit with [l4_thread_modify_sender_commit\(\)](#). Do not touch the UTCB between [l4_thread_modify_sender_start\(\)](#) and [l4_thread_modify_sender_commit\(\)](#).

See also

[l4_thread_modify_sender_add](#)
[l4_thread_modify_sender_commit](#)

Definition at line 1003 of file [thread.h](#).

12.103.3.9 `l4_thread_register_del_irq()`

```
l4_msgtag_t l4_thread_register_del_irq (
    l4_cap_idx_t thread,
    l4_cap_idx_t irq ) [inline]
```

Register an IRQ that will trigger upon deletion events.

Parameters

<i>thread</i>	Thread to register IRQ for.
<i>irq</i>	Capability selector for the IRQ object to be triggered.

Returns

System call return tag containing the return code.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

Definition at line 930 of file [thread.h](#).

12.103.3.10 l4_thread_stats_time()

```
l4_msgtag_t l4_thread_stats_time (
    l4_cap_idx_t thread,
    l4_kernel_clock_t * us ) [inline]
```

Get consumed time of thread in μ s.

Parameters

	<i>thread</i>	Thread to get the consumed time from.
out	<i>us</i>	Consumed time in μ s.

Returns

system call return tag

Definition at line 898 of file [thread.h](#).

12.103.3.11 l4_thread_switch()

```
l4_msgtag_t l4_thread_switch (
    l4_cap_idx_t to_thread ) [inline]
```

Switch to another thread (and donate the remaining time slice).

Parameters

<i>to_thread</i>	The thread to switch to.
------------------	--------------------------

Returns

system call return tag

Definition at line 889 of file [thread.h](#).

12.103.3.12 `l4_thread_vcpu_control()`

```
l4_msgtag_t l4_thread_vcpu_control (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state ) [inline]
```

Enable or disable the vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
<i>vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see l4_task_add_ku_mem()).

Returns

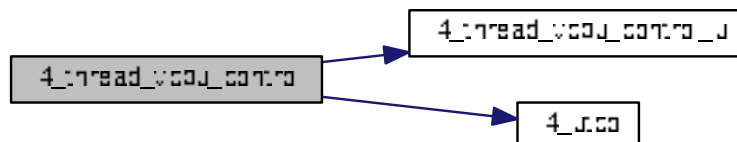
Syscall return tag.

This function enables the vCPU feature of the `thread` if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 947 of file [thread.h](#).

References [l4_thread_vcpu_control_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

12.103.3.13 `l4_thread_vcpu_control_ext()`

```
l4_msgtag_t l4_thread_vcpu_control_ext (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state ) [inline]
```

Enable or disable the extended vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the extended vCPU feature shall be enabled or disabled.
<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see l4_task_add_ku_mem()).

Returns

Systemcall result message tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of the `thread` if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

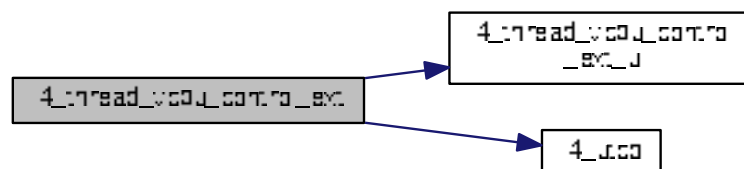
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 962 of file [thread.h](#).

References [l4_thread_vcpu_control_ext_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

12.103.3.14 `l4_thread_vcpu_control_ext_u()`

```

l4_msgtag_t l4_thread_vcpu_control_ext_u (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb ) [inline]
  
```

Enable or disable the extended vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the extended vCPU feature shall be enabled or disabled.
<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
<i>utcb</i>	UTCB to use for this operation.

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of `this thread` if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

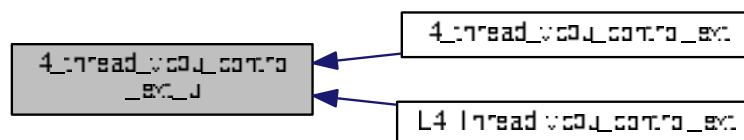
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 952 of file [thread.h](#).

Referenced by [l4_thread_vcpu_control_ext\(\)](#), and [L4::Thread::vcpu_control_ext\(\)](#).

Here is the caller graph for this function:



12.103.3.15 l4_thread_vcpu_control_u()

```

l4_msgtag_t l4_thread_vcpu_control_u (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb ) [inline]
  
```

Enable or disable the vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
<i>vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
<i>utcb</i>	UTCB to use for this operation.

Returns

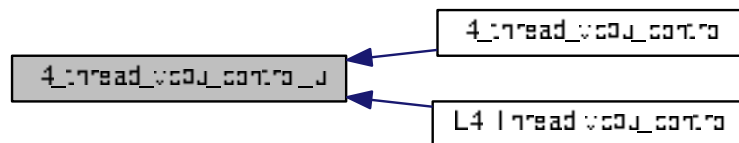
Syscall return tag.

This function enables the vCPU feature of *this* thread if *vcpu_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu_state* is 0. (Disable: optional, currently unsupported.)

Definition at line 937 of file [thread.h](#).

Referenced by [l4_thread_vcpu_control\(\)](#), and [L4::Thread::vcpu_control\(\)](#).

Here is the caller graph for this function:

12.103.3.16 `l4_thread_vcpu_resume_commit()`

```

l4_msgtag_t l4_thread_vcpu_resume_commit (
    l4_cap_idx_t thread,
    l4_msgtag_t tag ) [inline]
  
```

Commit vCPU resume.

Parameters

<i>thread</i>	Thread to be resumed, the invalid cap can be used for the current thread.
<i>tag</i>	Tag to use, returned by l4_thread_vcpu_resume_start()

Returns

System call result message tag. In extended vCPU mode and when the virtual interrupts are cleared, the return code 1 flags an incoming IPC message, whereas 0 indicates a VM exit. An error is returned upon:

- Insufficient rights on the given task capability (-L4_EPERM).
- Given task capability is invalid (-L4_ENOENT).
- A supplied mapping failed.

To resume into another address space the capability to the target task must be set in the vCPU-state, with all lower bits in the task capability cleared (see [L4_CAP_MASK](#)). The kernel adds the [L4_SYSF_SEND](#) flag to this field to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all bits are cleared again. To release a task use the different task capability or use an invalid capability with the [L4_SYSF_REPLY](#) flag set.

See also

[l4_vcpu_state_t](#)

Definition at line 910 of file [thread.h](#).

12.103.3.17 l4_thread_vcpu_resume_start()

```
l4_msgtag_t l4_thread_vcpu_resume_start (
    void ) [inline]
```

vCPU return from event handler.

Returns

Message tag to be used for [l4_sndfpage_add\(\)](#) and [l4_thread_vcpu_resume_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flex-pages using [l4_sndfpage_add\(\)](#).

Definition at line 904 of file [thread.h](#).

12.103.3.18 l4_thread_yield()

```
l4_msgtag_t l4_thread_yield (
    void ) [inline]
```

Yield current time slice.

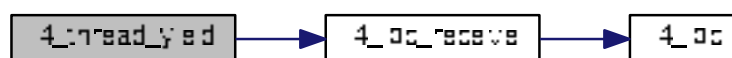
Returns

system call return tag

Definition at line 778 of file [thread.h](#).

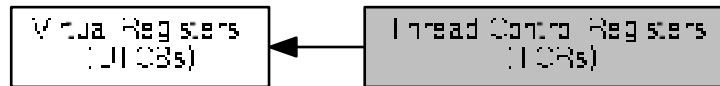
References [L4_INVALID_CAP](#), and [l4_ipc_receive\(\)](#).

Here is the call graph for this function:



12.104 Thread Control Registers (TCRs)

Collaboration diagram for Thread Control Registers (TCRs):



Data Structures

- struct [l4_thread_regs_t](#)
Encapsulation of the thread-control-register block of the UTCB.

Typedefs

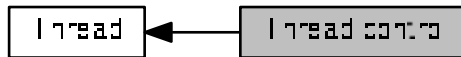
- typedef struct [l4_thread_regs_t](#) [l4_thread_regs_t](#)
Encapsulation of the thread-control-register block of the UTCB.

12.104.1 Detailed Description

12.105 Thread control

API for Thread Control method.

Collaboration diagram for Thread control:



Functions

- void [l4_thread_control_start](#) (void) [L4_NOTHROW](#)
Start a thread control API sequence.
- void [l4_thread_control_pager](#) ([l4_cap_idx_t](#) pager) [L4_NOTHROW](#)
Set the pager.
- void [l4_thread_control_exc_handler](#) ([l4_cap_idx_t](#) exc_handler) [L4_NOTHROW](#)
Set the exception handler.
- void [l4_thread_control_bind](#) ([l4_utcb_t](#) *thread_utcb, [l4_cap_idx_t](#) task) [L4_NOTHROW](#)
Bind the thread to a task.
- void [l4_thread_control_alien](#) (int on) [L4_NOTHROW](#)
Enable alien mode.
- void [l4_thread_control_ux_host_syscall](#) (int on) [L4_NOTHROW](#)
Enable pass through of native host (Linux) system calls.
- [l4_msgtag_t](#) [l4_thread_control_commit](#) ([l4_cap_idx_t](#) thread) [L4_NOTHROW](#)
Commit the thread control parameters.

12.105.1 Detailed Description

API for Thread Control method.

The thread control API provides access to almost any parameter of a thread object. The API is based on a single invocation of the thread object. However, because of the huge amount of parameters, the API provides a set of functions to set specific parameters of a thread and a commit function to commit the thread control call (see [l4_thread_control_commit\(\)](#)).

A thread control operation must always start with [l4_thread_control_start\(\)](#) and be committed with [l4_thread_control_commit\(\)](#). All other thread control parameter setter functions must be called between these two functions.

An example for a sequence of thread control API calls can be found below.

```

l4_utcb_t *u = l4_utcb();
l4_thread_control_start(u);
l4_thread_control_pager(u, pager_cap);
l4_thread_control_bind (u, thread_utcb, task);
l4_thread_control_commit(u, thread_cap);
  
```

12.105.2 Function Documentation

12.105.2.1 l4_thread_control_alien()

```
void l4_thread_control_alien (
    int on ) [inline]
```

Enable alien mode.

Parameters

<i>on</i>	Boolean value defining the state of the feature.
-----------	--

Alien mode means the thread is not allowed to invoke [L4](#) kernel objects directly and it is also not allowed to allocate FPU state. All those operations result in an exception IPC that gets sent through the pager capability. The responsible pager can then selectively allow an object invocation or allocate FPU state for the thread.

This feature can be used to attach a debugger to a thread and trace all object invocations.

Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [868](#) of file [thread.h](#).

12.105.2.2 l4_thread_control_bind()

```
void l4_thread_control_bind (
    l4_utcb_t * thread_utcb,
    l4_cap_idx_t task ) [inline]
```

Bind the thread to a task.

Parameters

<i>thread_utcb</i>	The address of the UTCB in the target task.
<i>task</i>	The target task of the thread.

Binding a thread to a task has the effect that the thread afterwards executes code within that task and has access to the resources visible within that task.

Note

There should not be more than one thread use a UTCB to prevent data corruption.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 862 of file [thread.h](#).

12.105.2.3 l4_thread_control_commit()

```
l4_msgtag_t l4_thread_control_commit (
    l4_cap_idx_t thread ) [inline]
```

Commit the thread control parameters.

Parameters

<i>thread</i>	Capability selector of target thread to commit to.
---------------	--

Returns

system call return tag

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 880 of file [thread.h](#).

12.105.2.4 l4_thread_control_exc_handler()

```
void l4_thread_control_exc_handler (
    l4_cap_idx_t exc_handler ) [inline]
```

Set the exception handler.

Parameters

<i>exc_handler</i>	Capability selector invoked to send an exception IPC.
--------------------	---

Note

The exception-handler capability selector is interpreted in the task the thread is bound to (executes in).

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 855 of file [thread.h](#).

12.105.2.5 l4_thread_control_pager()

```
void l4_thread_control_pager (
    l4_cap_idx_t pager ) [inline]
```

Set the pager.

Parameters

<i>pager</i>	Capability selector invoked to send a page-fault IPC.
--------------	---

Note

The pager capability selector is interpreted in the task the thread is bound to (executes in).

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 849 of file [thread.h](#).

12.105.2.6 l4_thread_control_start()

```
void l4_thread_control_start (
    void ) [inline]
```

Start a thread control API sequence.

This function starts a sequence of thread control API functions. After this functions any of following functions may be called in any order.

- [l4_thread_control_pager\(\)](#)
- [l4_thread_control_exc_handler\(\)](#)
- [l4_thread_control_bind\(\)](#)
- [l4_thread_control_alien\(\)](#)
- [l4_thread_control_ux_host_syscall\(\)](#) (Fiasco-UX only)

To commit the changes to the thread [l4_thread_control_commit\(\)](#) must be called in the end.

Note

The thread control API calls store the parameters for the thread in the UTCB of the caller, this means between [l4_thread_control_start\(\)](#) and [l4_thread_control_commit\(\)](#) no functions that modify the UTCB contents must be called.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 843 of file [thread.h](#).

12.105.2.7 l4_thread_control_ux_host_syscall()

```
void l4_thread_control_ux_host_syscall (  
    int on ) [inline]
```

Enable pass through of native host (Linux) system calls.

Parameters

<i>on</i>	Boolean value defining the state of the feature.
-----------	--

Precondition

Running on Fiasco-UX

This enables the thread to do host system calls. This feature is only available in Fiasco-UX and ignored in other environments.

Definition at line [874](#) of file [thread.h](#).

12.106 Timeouts

All kinds of timeouts and time related functions.

Collaboration diagram for Timeouts:



Data Structures

- struct [l4_timeout_s](#)
Basic timeout specification.
- union [l4_timeout_t](#)
Timeout pair.

Macros

- #define [L4_IPC_TIMEOUT_0](#) (([l4_timeout_s](#)){0x0400})
Timeout constants.
- #define [L4_IPC_TIMEOUT_NEVER](#) (([l4_timeout_s](#)){0})
never timeout
- #define [L4_IPC_NEVER_INITIALIZER](#) {0}
never timeout, init
- #define [L4_IPC_NEVER](#) (([l4_timeout_t](#)){0})
never timeout
- #define [L4_IPC_RECV_TIMEOUT_0](#) (([l4_timeout_t](#)){0x00000400})
0 receive timeout
- #define [L4_IPC_SEND_TIMEOUT_0](#) (([l4_timeout_t](#)){0x04000000})
0 send timeout
- #define [L4_IPC_BOTH_TIMEOUT_0](#) (([l4_timeout_t](#)){0x04000400})
0 receive and send timeout

Typedefs

- typedef struct [l4_timeout_s](#) [l4_timeout_s](#)
Basic timeout specification.
- typedef union [l4_timeout_t](#) [l4_timeout_t](#)
Timeout pair.

Enumerations

- enum [l4_timeout_abs_validity](#)
Intervals of validity for absolute timeouts
Times are actually 2^x values (e.g.

Functions

- [l4_timeout_s l4_timeout_rel](#) (unsigned man, unsigned exp) [L4_NOTHROW](#)
Get relative timeout consisting of mantissa and exponent.
- [l4_timeout_t l4_ipc_timeout](#) (unsigned snd_man, unsigned snd_exp, unsigned rcv_man, unsigned rcv_exp) [L4_NOTHROW](#)
Convert explicit timeout values to [l4_timeout_t](#) type.
- [l4_timeout_t l4_timeout](#) ([l4_timeout_s](#) snd, [l4_timeout_s](#) rcv) [L4_NOTHROW](#)
Combine send and receive timeout in a timeout.
- void [l4_snd_timeout](#) ([l4_timeout_s](#) snd, [l4_timeout_t](#) *to) [L4_NOTHROW](#)
Set send timeout in given to timeout.
- void [l4_rcv_timeout](#) ([l4_timeout_s](#) rcv, [l4_timeout_t](#) *to) [L4_NOTHROW](#)
Set receive timeout in given to timeout.
- [l4_kernel_clock_t l4_timeout_rel_get](#) ([l4_timeout_s](#) to) [L4_NOTHROW](#)
Get clock value of out timeout.
- unsigned [l4_timeout_is_absolute](#) ([l4_timeout_s](#) to) [L4_NOTHROW](#)
Return whether the given timeout is absolute or not.
- [l4_kernel_clock_t l4_timeout_get](#) ([l4_kernel_clock_t](#) cur, [l4_timeout_s](#) to) [L4_NOTHROW](#)
Get clock value for a clock + a timeout.
- [l4_timeout_s l4_timeout_abs](#) ([l4_kernel_clock_t](#) pint, int br) [L4_NOTHROW](#)
Set an absolute timeout.
- unsigned [l4_utcb_mr64_idx](#) (unsigned idx) [L4_NOTHROW](#)
Get index into 64bit message registers alias from native-sized index.

12.106.1 Detailed Description

All kinds of timeouts and time related functions.

12.106.2 Macro Definition Documentation

12.106.2.1 L4_IPC_TIMEOUT_0

```
#define L4_IPC_TIMEOUT_0 ((l4\_timeout\_s) {0x0400})
```

Timeout constants.

0 timeout

Definition at line 77 of file [__timeout.h](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout\(\)](#).

12.106.3 Typedef Documentation

12.106.3.1 l4_timeout_s

```
typedef struct l4_timeout_s l4_timeout_s
```

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m \cdot 2^e$).

The timeout can also specify an absolute point in time (bit 16 == 1).

12.106.3.2 l4_timeout_t

```
typedef union l4_timeout_t l4_timeout_t
```

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

12.106.4 Enumeration Type Documentation

12.106.4.1 l4_timeout_abs_validity

```
enum l4_timeout_abs_validity
```

Intervals of validity for absolute timeouts

Times are actually 2^x values (e.g.

2ms -> 2048μs)

Definition at line 92 of file [__timeout.h](#).

12.106.5 Function Documentation

12.106.5.1 l4_ipc_timeout()

```
l4_timeout_t l4_ipc_timeout (
    unsigned snd_man,
    unsigned snd_exp,
    unsigned rcv_man,
    unsigned rcv_exp ) [inline]
```

Convert explicit timeout values to [l4_timeout_t](#) type.

Parameters

<i>snd_man</i>	Mantissa of send timeout.
<i>snd_exp</i>	Exponent of send timeout.
<i>rcv_man</i>	Mantissa of receive timeout.
<i>rcv_exp</i>	Exponent of receive timeout.

Definition at line 210 of file [__timeout.h](#).

References [l4_timeout_t::p](#), [l4_timeout_t::rcv](#), [l4_timeout_t::snd](#), and [l4_timeout_s::t](#).

12.106.5.2 l4_rcv_timeout()

```
void l4_rcv_timeout (
    l4_timeout_s rcv,
    l4_timeout_t * to ) [inline]
```

Set receive timeout in given to timeout.

Parameters

<i>rcv</i>	Receive timeout
------------	-----------------

Return values

<i>to</i>	L4 timeout
-----------	----------------------------

Definition at line 238 of file [__timeout.h](#).

12.106.5.3 l4_snd_timeout()

```
void l4_snd_timeout (
    l4_timeout_s snd,
    l4_timeout_t * to ) [inline]
```

Set send timeout in given to timeout.

Parameters

<i>snd</i>	Send timeout
------------	--------------

Return values

<i>to</i>	L4 timeout
-----------	----------------------------

Definition at line 231 of file [__timeout.h](#).

12.106.5.4 l4_timeout()

```
l4_timeout_t l4_timeout (
    l4_timeout_s snd,
    l4_timeout_s rcv ) [inline]
```

Combine send and receive timeout in a timeout.

Parameters

<i>snd</i>	Send timeout
<i>rcv</i>	Receive timeout

Returns

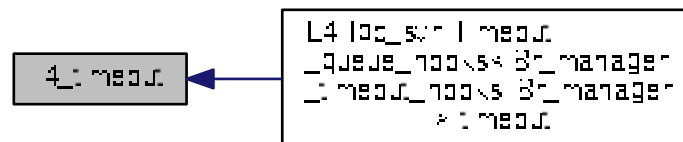
[L4](#) timeout

Definition at line 221 of file [__timeout.h](#).

References [l4_timeout_t::p](#), [l4_timeout_t::rcv](#), [l4_timeout_t::snd](#), and [l4_timeout_s::t](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout\(\)](#).

Here is the caller graph for this function:



12.106.5.5 l4_timeout_abs()

```
l4_timeout_s l4_timeout_abs (
    l4_kernel_clock_t pint,
    int br ) [inline]
```

Set an absolute timeout.

Parameters

<i>pint</i>	Point in time in clocks
<i>br</i>	The buffer register the timeout shall be placed in. (

Note

On 32bit architectures the timeout needs two consecutive buffers.)
 The absolute timeout value will be placed into the buffer register *br* of the current thread.

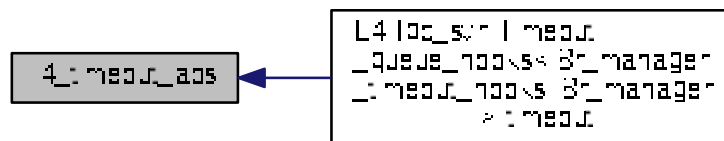
Returns

timeout value

Definition at line 383 of file [utcb.h](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout\(\)](#).

Here is the caller graph for this function:



12.106.5.6 l4_timeout_get()

```
l4_kernel_clock_t l4_timeout_get (
    l4_kernel_clock_t cur,
    l4_timeout_s to ) [inline]
```

Get clock value for a clock + a timeout.

Parameters

<i>cur</i>	Clock value
<i>to</i>	L4 timeout

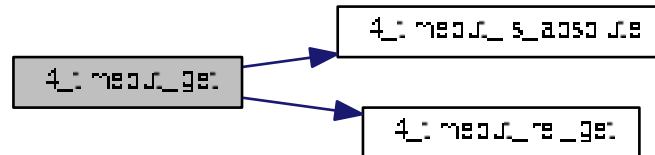
Returns

Clock sum

Definition at line 268 of file [__timeout.h](#).

References [l4_timeout_is_absolute\(\)](#), and [l4_timeout_rel_get\(\)](#).

Here is the call graph for this function:



12.106.5.7 l4_timeout_is_absolute()

```
unsigned l4_timeout_is_absolute (
    l4_timeout_s to ) [inline]
```

Return whether the given timeout is absolute or not.

Parameters

<i>to</i>	L4 timeout
-----------	------------

Returns

!= 0 if absolute, 0 if relative

Definition at line 261 of file [__timeout.h](#).

References [l4_timeout_s::t](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:



12.106.5.8 l4_timeout_rel()

```
l4_timeout_s l4_timeout_rel (
    unsigned man,
    unsigned exp ) [inline]
```

Get relative timeout consisting of mantissa and exponent.

Parameters

<i>man</i>	Mantissa of timeout
<i>exp</i>	Exponent of timeout

Returns

timeout value

Definition at line 245 of file [__timeout.h](#).

12.106.5.9 l4_timeout_rel_get()

```
l4_kernel_clock_t l4_timeout_rel_get (
    l4_timeout_s to ) [inline]
```

Get clock value of out timeout.

Parameters

<i>to</i>	L4 timeout
-----------	------------

Returns

Clock value

Definition at line 252 of file [__timeout.h](#).

References [l4_timeout_s::t](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:



12.106.5.10 l4_utcb_mr64_idx()

```
unsigned l4_utcb_mr64_idx (
    unsigned idx ) [inline]
```

Get index into 64bit message registers alias from native-sized index.

Parameters

<i>idx</i>	Index to native-sized message register
------------	--

Returns

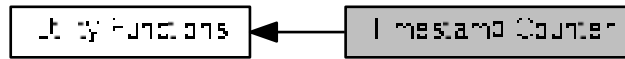
Index to 64bit message register alias

Definition at line [386](#) of file [utcb.h](#).

References [__END_DECLS](#).

12.107 Timestamp Counter

Collaboration diagram for Timestamp Counter:



Files

- file [rdtsc.h](#)
time stamp counter related functions
- file [rdtsc.h](#)
time stamp counter related functions

Macros

- `#define L4_TSC_INIT_AUTO 0`
Automatic init.
- `#define L4_TSC_INIT_KERNEL 1`
Initialized by kernel.
- `#define L4_TSC_INIT_CALIBRATE 2`
Initialized by user-level.
- `#define L4_TSC_INIT_AUTO 0`
Automatic init.
- `#define L4_TSC_INIT_KERNEL 1`
Initialized by kernel.
- `#define L4_TSC_INIT_CALIBRATE 2`
Initialized by user-level.

Functions

- `l4_cpu_time_t l4_rdtsc (void)`
Read current value of CPU-internal time stamp counter.
- `l4_uint32_t l4_rdtsc_32 (void)`
Read the lest significant 32 bit of the TSC.
- `l4_cpu_time_t l4_rdpmc (int nr)`
Return current value of CPU-internal performance measurement counter.
- `l4_uint32_t l4_rdpmc_32 (int nr)`
Return the least significant 32 bit of a performance counter.
- `l4_uint64_t l4_tsc_to_ns (l4_cpu_time_t tsc)`
Convert time stamp to ns value.
- `l4_uint64_t l4_tsc_to_us (l4_cpu_time_t tsc)`

Convert time stamp into micro seconds value.

- void [l4_tsc_to_s_and_ns](#) ([l4_cpu_time_t](#) tsc, [l4_uint32_t](#) *s, [l4_uint32_t](#) *ns)

Convert time stamp to s.ns value.

- [l4_cpu_time_t](#) [l4_ns_to_tsc](#) ([l4_uint64_t](#) ns)

Convert nano seconds into CPU ticks.

- void [l4_busy_wait_ns](#) ([l4_uint64_t](#) ns)

Wait busy for a small amount of time.

- void [l4_busy_wait_us](#) ([l4_uint64_t](#) us)

Wait busy for a small amount of time.

- [l4_uint32_t](#) [l4_calibrate_tsc](#) ([l4_kernel_info_t](#) *kip)

Calibrate scalars for time stamp calculations.

- [l4_uint32_t](#) [l4_tsc_init](#) (int constraint, [l4_kernel_info_t](#) *kip)

Initialitze scaler for TSC calicatlions.

- [l4_uint32_t](#) [l4_get_hz](#) (void)

Get CPU frequency in Hz.

12.107.1 Detailed Description

12.107.2 Function Documentation

12.107.2.1 [l4_busy_wait_ns\(\)](#)

```
void l4_busy_wait_ns (
    l4\_uint64\_t ns ) [inline]
```

Wait busy for a small amount of time.

Parameters

ns	nano seconds to wait
--------------------	----------------------

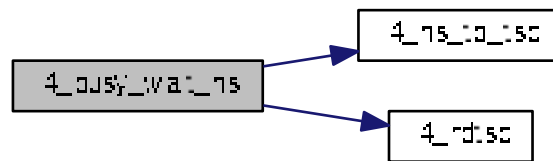
Attention

Not intendet for any use!

Definition at line [317](#) of file [rdtsc.h](#).

References [l4_ns_to_tsc\(\)](#), and [l4_rdtsc\(\)](#).

Here is the call graph for this function:



12.107.2.2 l4_busy_wait_us()

```
void l4_busy_wait_us (
    l4_uint64_t us ) [inline]
```

Wait busy for a small amount of time.

Parameters

<i>us</i>	micro seconds to wait
-----------	-----------------------

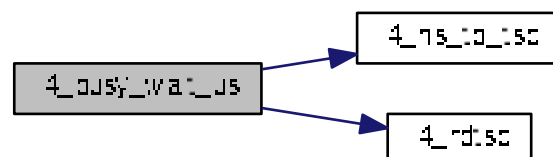
Attention

Not intended for any use!

Definition at line 327 of file [rdtsc.h](#).

References [EXTERN_C_END](#), [l4_ns_to_tsc\(\)](#), and [l4_rdtsc\(\)](#).

Here is the call graph for this function:



12.107.2.3 l4_calibrate_tsc()

```
l4_uint32_t l4_calibrate_tsc (
    l4_kernel_info_t * kip ) [inline]
```

Calibrate scalers for time stamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. This test uses channel 0 of the PIT (i8254) or the kernel KIP, depending on availability. Just calls `l4_tsc_init(L4_TSC_INIT_AUTO)`.

Examples:

[examples/sys/aliens/main.c](#).

Definition at line 179 of file [rdtsc.h](#).

References [l4_tsc_init\(\)](#), and [L4_TSC_INIT_AUTO](#).

Here is the call graph for this function:



12.107.2.4 l4_get_hz()

```
l4_uint32_t l4_get_hz (
    void )
```

Get CPU frequency in Hz.

Returns

frequency in Hz

12.107.2.5 l4_ns_to_tsc()

```
l4_cpu_time_t l4_ns_to_tsc (
    l4_uint64_t ns ) [inline]
```

Convert nano seconds into CPU ticks.

Parameters

<i>ns</i>	nano seconds
-----------	--------------

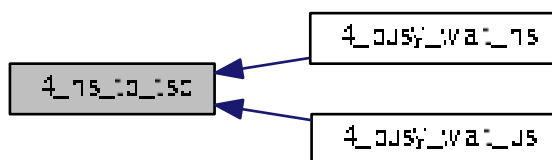
Returns

CPU ticks

Definition at line 303 of file [rdtsc.h](#).

Referenced by [l4_busy_wait_ns\(\)](#), and [l4_busy_wait_us\(\)](#).

Here is the caller graph for this function:



12.107.2.6 l4_rdpmc()

```
l4_cpu_time_t l4_rdpmc (
    int nr ) [inline]
```

Return current value of CPU-internal performance measurement counter.

Parameters

<i>nr</i>	Number of counter (0 or 1)
-----------	----------------------------

Returns

64-bit PMC

Definition at line 205 of file [rdtsc.h](#).

12.107.2.7 l4_rdpmc_32()

```
l4_uint32_t l4_rdpmc_32 (
    int nr ) [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 227 of file [rdtsc.h](#).

12.107.2.8 l4_rdtsc()

```
l4_cpu_time_t l4_rdtsc (
    void ) [inline]
```

Read current value of CPU-internal time stamp counter.

Returns

64-bit time stamp

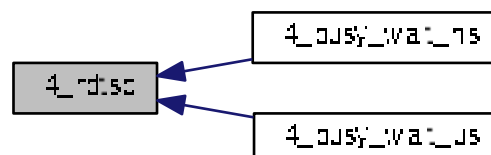
Examples:

[examples/sys/aliens/main.c](#).

Definition at line 185 of file [rdtsc.h](#).

Referenced by [l4_busy_wait_ns\(\)](#), and [l4_busy_wait_us\(\)](#).

Here is the caller graph for this function:



12.107.2.9 l4_rdtsc_32()

```
l4_uint32_t l4_rdtsc_32 (
    void ) [inline]
```

Read the lest significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 246 of file [rdtsc.h](#).

12.107.2.10 l4_tsc_init()

```
l4_uint32_t l4_tsc_init (
    int constraint,
    l4_kernel_info_t * kip )
```

Initialitze scaler for TSC calicaltions.

Initialize the scalers needed by [l4_tsc_to_ns\(\)](#)/[l4_ns_to_tsc\(\)](#) and so on. Current versions of Fiasco export these scalers from kernel into userland. The programmer may decide whether he allows to use these scalers or if an calibration should be performed.

Parameters

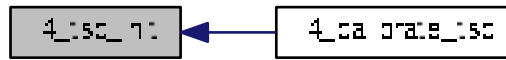
<i>constraint</i>	<p>programmers constraint:</p> <ul style="list-style-type: none"> • L4_TSC_INIT_AUTO if the kernel exports the scalers then use them. If not, perform calibration using channel 0 of the PIT (i8254). The latter case may lead into short (unpredictable) periods where interrupts are disabled. • L4_TSC_INIT_KERNEL depend on retrieving the scalers from kernel. If the scalers are not available, return 0. • L4_TSC_INIT_CALIBRATE Ignore possible scalers exported by the scaler, instead insist on calibration using the PIT.
<i>kip</i>	KIP pointer

Returns

0 on error (no scalers exported by kernel, calibrating failed ...) otherwise returns ($2^{32} / (\text{tsc per } \mu\text{sec})$). This value has the same semantics as the value returned by the `calibrate_delay_loop()` function of the Linux kernel.

Referenced by [l4_calibrate_tsc\(\)](#).

Here is the caller graph for this function:



12.107.2.11 l4_tsc_to_ns()

```
l4_uint64_t l4_tsc_to_ns (
    l4_cpu_time_t tsc ) [inline]
```

Convert time stamp to ns value.

Parameters

<code>tsc</code>	time value in CPU ticks
------------------	-------------------------

Returns

time value in ns

Examples:

[examples/sys/aliens/main.c](#).

Definition at line 260 of file [rdtsc.h](#).

12.107.2.12 l4_tsc_to_s_and_ns()

```
void l4_tsc_to_s_and_ns (
    l4_cpu_time_t tsc,
    l4_uint32_t * s,
    l4_uint32_t * ns ) [inline]
```

Convert time stamp to s.ns value.

Parameters

<code>tsc</code>	time value in CPU ticks
------------------	-------------------------

Return values

<i>s</i>	seconds
<i>ns</i>	nano seconds

Definition at line 288 of file [rdtsc.h](#).

12.107.2.13 l4_tsc_to_us()

```
l4_uint64_t l4_tsc_to_us (
    l4_cpu_time_t tsc ) [inline]
```

Convert time stamp into micro seconds value.

Parameters

<i>tsc</i>	time value in CPU ticks
------------	-------------------------

Returns

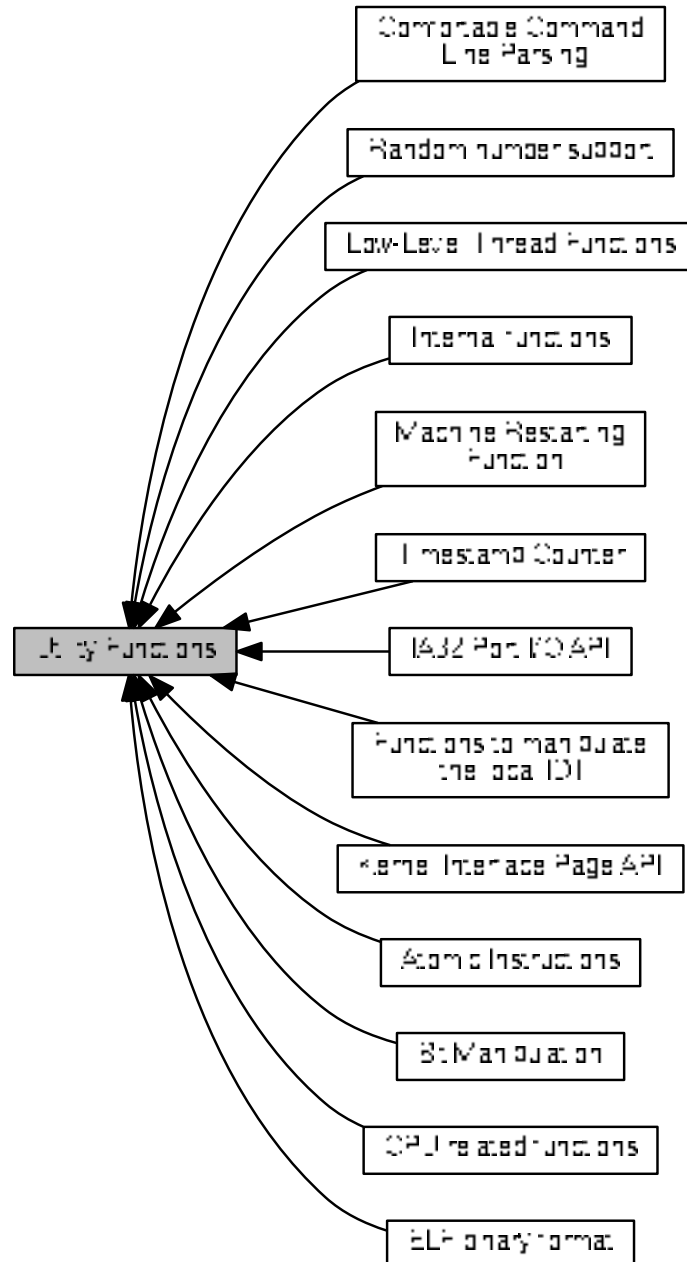
time value in micro seconds

Definition at line 274 of file [rdtsc.h](#).

12.108 Utility Functions

Utilities, generic file.

Collaboration diagram for Utility Functions:



Modules

- [Atomic Instructions](#)

- [Bit Manipulation](#)
- [CPU related functions](#)
- [Comfortable Command Line Parsing](#)
- [ELF binary format](#)

Functions and types related to ELF binaries.

- [Functions to manipulate the local IDT](#)
- [IA32 Port I/O API](#)
- [Internal functions](#)
- [Kernel Interface Page API](#)
- [Low-Level Thread Functions](#)
- [Machine Restarting Function](#)
- [Random number support](#)
- [Timestamp Counter](#)

Files

- file [rand.h](#)

Simple Pseudo-Random Number Generator.

Functions

- void [l4_sleep_forever](#) (void) [L4_NOTHROW](#)
Go sleep and never wake up.
- long [l4util_splitlog2_hdl](#) ([l4_addr_t](#) start, [l4_addr_t](#) end, long(*handler)([l4_addr_t](#) s, [l4_addr_t](#) e, int log2size))
Split a range into log2 base and size aligned chunks.
- [l4_addr_t](#) [l4util_splitlog2_size](#) ([l4_addr_t](#) start, [l4_addr_t](#) end)
Return log2 base and size aligned length of a range.
- [l4_timeout_s](#) [l4util_micros2l4to](#) (unsigned int mus) [L4_NOTHROW](#)
Calculate l4 timeouts.
- void [l4_sleep](#) (int ms) [L4_NOTHROW](#)
Suspend thread for a period of ms milliseconds.
- void [l4_usleep](#) (int us) [L4_NOTHROW](#)
Suspend thread for a period of us microseconds.
- void [l4_touch_ro](#) (const void *addr, unsigned size) [L4_NOTHROW](#)
Touch data area to force mapping (read-only)
- void [l4_touch_rw](#) (const void *addr, unsigned size) [L4_NOTHROW](#)
Touch data areas to force mapping (read-write)

12.108.1 Detailed Description

Utilities, generic file.

12.108.2 Function Documentation

12.108.2.1 [l4_sleep\(\)](#)

```
void l4_sleep (
    int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

<i>ms</i>	Time in milliseconds
-----------	----------------------

12.108.2.2 `l4_touch_ro()`

```
void l4_touch_ro (
    const void * addr,
    unsigned size ) [inline]
```

Touch data area to force mapping (read-only)

Parameters

<i>addr</i>	Start of memory area to touch.
<i>size</i>	Size of area to touch.

Examples:

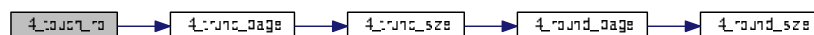
[examples/sys/singlestep/main.c](#).

Definition at line 94 of file [util.h](#).

References [L4_PAGESIZE](#), and [l4_trunc_page\(\)](#).

Referenced by [l4_sleep_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.108.2.3 l4_touch_rw()

```
void l4_touch_rw (
    const void * addr,
    unsigned size ) [inline]
```

Touch data areas to force mapping (read-write)

Parameters

<i>addr</i>	Start of memory area to touch.
<i>size</i>	Size of area to touch.

Examples:

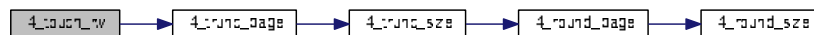
[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 107 of file [util.h](#).

References [EXTERN_C_END](#), [L4_PAGESIZE](#), and [l4_trunc_page\(\)](#).

Referenced by [l4_sleep_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.108.2.4 l4_usleep()

```
void l4_usleep (
    int us )
```

Suspend thread for a period of *us* microseconds.

Parameters

<i>us</i>	Time in microseconds
-----------	----------------------

WARNING: This function is mostly bogus since the timer resolution of current [L4](#) implementations is about 1ms!

12.108.2.5 l4util_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
    unsigned int mus )
```

Calculate l4 timeouts.

Parameters

<i>mus</i>	time in microseconds. Special cases: <ul style="list-style-type: none"> • 0 -> timeout 0 • ~0U -> timeout NEVER
------------	---

Returns

the corresponding l4_timeout value

12.108.2.6 l4util_splitlog2_hdl()

```
long l4util_splitlog2_hdl (
    l4_addr_t start,
    l4_addr_t end,
    long(*) (l4_addr_t s, l4_addr_t e, int log2size) handler ) [inline]
```

Split a range into log2 base and size aligned chunks.

Parameters

<i>start</i>	Start of range
<i>end</i>	End of range (inclusive) (e.g. 2-4 is len 3)
<i>handler</i>	Handler function that is called with start and end (both inclusive) of the chunk. On success, the handler must return 0, if it returns !=0 the function will immediately return with the return code of the handler.

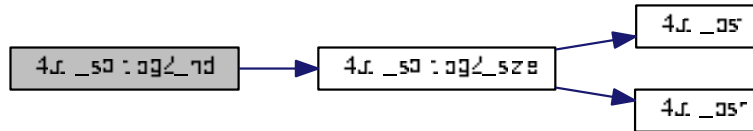
Returns

0 on success, != 0 otherwise

Definition at line 53 of file [splitlog2.h](#).

References [L4_EINVAL](#), and [l4util_splitlog2_size\(\)](#).

Here is the call graph for this function:



12.108.2.7 l4util_splitlog2_size()

```
l4_addr_t l4util_splitlog2_size (
    l4_addr_t start,
    l4_addr_t end ) [inline]
```

Return log2 base and size aligned length of a range.

Parameters

<i>start</i>	Start of range
<i>end</i>	End of range (inclusive) (e.g. 2-4 is len 3)

Returns

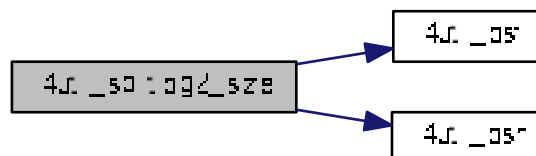
length of elements in log2size (length is $1 < \log_2 \text{size}$)

Definition at line 72 of file [splitlog2.h](#).

References [l4util_bsf\(\)](#), and [l4util_bsr\(\)](#).

Referenced by [l4util_splitlog2_hdl\(\)](#).

Here is the call graph for this function:



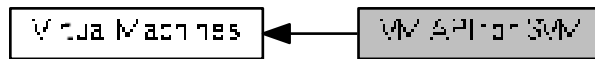
Here is the caller graph for this function:



12.109 VM API for SVM

Virtual machine API for SVM.

Collaboration diagram for VM API for SVM:



Data Structures

- struct [l4_vm_svm_vmcb_control_area](#)
VMCB structure for SVM VMs.
- struct [l4_vm_svm_vmcb_state_save_area_seg](#)
State save area segment selector struct.
- struct [l4_vm_svm_vmcb_state_save_area](#)
State save area structure for SVM VMs.
- struct [l4_vm_svm_vmcb_t](#)
Control structure for SVM VMs.

Typedefs

- typedef struct [l4_vm_svm_vmcb_control_area](#) [l4_vm_svm_vmcb_control_area_t](#)
VMCB structure for SVM VMs.
- typedef struct [l4_vm_svm_vmcb_state_save_area_seg](#) [l4_vm_svm_vmcb_state_save_area_seg_t](#)
State save area segment selector struct.
- typedef struct [l4_vm_svm_vmcb_state_save_area](#) [l4_vm_svm_vmcb_state_save_area_t](#)
State save area structure for SVM VMs.
- typedef struct [l4_vm_svm_vmcb_t](#) [l4_vm_svm_vmcb_t](#)
Control structure for SVM VMs.

12.109.1 Detailed Description

Virtual machine API for SVM.

12.110 VM API for TZ

Virtual Machine API for ARM TrustZone.

Collaboration diagram for VM API for TZ:



Data Structures

- struct [l4_vm_tz_state](#)
state structure for TrustZone VMs

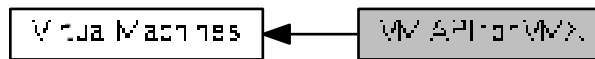
12.110.1 Detailed Description

Virtual Machine API for ARM TrustZone.

12.111 VM API for VMX

Virtual machine API for VMX.

Collaboration diagram for VM API for VMX:



Enumerations

- enum `L4_vm_vmx_caps_regs` {
`L4_VM_VMX_BASIC_REG` = 0, `L4_VM_VMX_TRUE_PINBASED_CTLS_REG` = 1, `L4_VM_VMX_TRUE_PROCBASED_CTLS_REG` = 2, `L4_VM_VMX_TRUE_EXIT_CTLS_REG` = 3,
`L4_VM_VMX_TRUE_ENTRY_CTLS_REG` = 4, `L4_VM_VMX_MISC_REG` = 5, `L4_VM_VMX_CR0_FIXED0_REG` = 6, `L4_VM_VMX_CR0_FIXED1_REG` = 7,
`L4_VM_VMX_CR4_FIXED0_REG` = 8, `L4_VM_VMX_CR4_FIXED1_REG` = 9, `L4_VM_VMX_VMCS_ENUM_REG` = 0xa, `L4_VM_VMX_PROCBASED_CTLS2_REG` = 0xb,
`L4_VM_VMX_EPT_VPID_CAP_REG` = 0xc, `L4_VM_VMX_NUM_CAPS_REGS` }
Exported VMX capability registers.
- enum `L4_vm_vmx_dfl1_regs` {
`L4_VM_VMX_PINBASED_CTLS_DFL1_REG` = 0x1, `L4_VM_VMX_PROCBASED_CTLS_DFL1_REG` = 0x2, `L4_VM_VMX_EXIT_CTLS_DFL1_REG` = 0x3, `L4_VM_VMX_ENTRY_CTLS_DFL1_REG` = 0x4,
`L4_VM_VMX_NUM_DFL1_REGS` }
Exported VMX capability registers (default to 1 bits).
- enum {
`L4_VM_VMX_VMCS_CR2` = 0x683e, `L4_VM_VMX_VMCS_XCR0` = 0x2840, `L4_VM_VMX_VMCS_MSR_SYSCALL_MASK` = 0x2842, `L4_VM_VMX_VMCS_MSR_LSTAR` = 0x2844,
`L4_VM_VMX_VMCS_MSR_CSTAR` = 0x2846, `L4_VM_VMX_VMCS_MSR_TSC_AUX` = 0x2848, `L4_VM_VMX_VMCS_MSR_STAR` = 0x284a, `L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE` = 0x284c }
Additional (virtual) VMCS fields.

Functions

- `l4_uint64_t l4_vm_vmx_get_caps` (void const *vcpu_state, unsigned cap_msr) `L4_NOTHROW`
Get a capability register for VMX.
- `l4_uint32_t l4_vm_vmx_get_caps_default1` (void const *vcpu_state, unsigned cap_msr) `L4_NOTHROW`
Get a default to one capability register for VMX.
- unsigned `l4_vm_vmx_field_len` (unsigned field) `L4_NOTHROW`
Return length in bytes of a VMCS field.
- unsigned `l4_vm_vmx_field_order` (unsigned field) `L4_NOTHROW`
Return length in power of two (bytes) of a VMCS field.
- void `l4_vm_vmx_clear` (void *vmcs, void *user_vmcs) `L4_NOTHROW`
Saves cached state from the kernel VMCS to the user VMCS.
- void `l4_vm_vmx_ptr_load` (void *vmcs, void *user_vmcs) `L4_NOTHROW`

- Loads the user_vmcs as the current VMCS.*
- [l4_uint32_t l4_vm_vmx_get_cr2_index](#) (void const *vmcs) [L4_NOTHROW](#)
Get the VMCS field index of the virtual CR2 register.
- [l4_umword_t l4_vm_vmx_read_nat](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a natural width VMCS field.
- [l4_uint16_t l4_vm_vmx_read_16](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 16bit VMCS field.
- [l4_uint32_t l4_vm_vmx_read_32](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 32bit VMCS field.
- [l4_uint64_t l4_vm_vmx_read_64](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 64bit VMCS field.
- [l4_uint64_t l4_vm_vmx_read](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read any VMCS field.
- void [l4_vm_vmx_write_nat](#) (void *vmcs, unsigned field, [l4_umword_t](#) val) [L4_NOTHROW](#)
Write to a natural width VMCS field.
- void [l4_vm_vmx_write_16](#) (void *vmcs, unsigned field, [l4_uint16_t](#) val) [L4_NOTHROW](#)
Write to a 16bit VMCS field.
- void [l4_vm_vmx_write_32](#) (void *vmcs, unsigned field, [l4_uint32_t](#) val) [L4_NOTHROW](#)
Write to a 32bit VMCS field.
- void [l4_vm_vmx_write_64](#) (void *vmcs, unsigned field, [l4_uint64_t](#) val) [L4_NOTHROW](#)
Write to a 64bit VMCS field.
- void [l4_vm_vmx_write](#) (void *vmcs, unsigned field, [l4_uint64_t](#) val) [L4_NOTHROW](#)
Write to an arbitrary VMCS field.

12.111.1 Detailed Description

Virtual machine API for VMX.

12.111.2 Enumeration Type Documentation

12.111.2.1 anonymous enum

anonymous enum

Additional (virtual) VMCS fields.

The VMCS offsets defined here are actually not in the hardware VMCS. However our VMMs run in user mode and need to have access to certain registers available in kernel mode only. So we put them into our version of the VMCS.

Enumerator

L4_VM_VMX_VMCS_CR2	VMCS offset for CR2. Note You usually need to check this value against the value you get from l4_vm_vmx_get_cr2_index() to make sure you are running on a compatible kernel.
L4_VM_VMX_VMCS_XCR0	VMCS offset of extended control register XCR0.
L4_VM_VMX_VMCS_MSR_SYSCALL_MASK	VMCS offset of system call flag mask MSR.
L4_VM_VMX_VMCS_MSR_LSTAR	VMCS offset of IA32e mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_CSTAR	VMCS offset of IA32 mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_TSC_AUX	VMCS offset of auxiliary TSC signature MSR.
L4_VM_VMX_VMCS_MSR_STAR	VMCS offset of system call target address MSR.
L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE	VMCS offset of GS base address swap target MSR.

Definition at line 105 of file [__vm-vmx.h](#).

12.111.2.2 L4_vm_vmx_caps_regs

enum [L4_vm_vmx_caps_regs](#)

Exported VMX capability registers.

Enumerator

L4_VM_VMX_BASIC_REG	Basic VMX capabilities.
L4_VM_VMX_TRUE_PINBASED_CTLN_REG	True pin-based control caps.
L4_VM_VMX_TRUE_PROCBASED_CTLN_REG	True processor based control caps.
L4_VM_VMX_TRUE_EXIT_CTLN_REG	True exit control caps.
L4_VM_VMX_TRUE_ENTRY_CTLN_REG	True entry control caps.
L4_VM_VMX_MISC_REG	Misc caps.
L4_VM_VMX_CR0_FIXED0_REG	Fixed to 0 bits of CR0.
L4_VM_VMX_CR0_FIXED1_REG	Fixed to 1 bits of CR0.
L4_VM_VMX_CR4_FIXED0_REG	Fixed to 0 bits of CR4.
L4_VM_VMX_CR4_FIXED1_REG	Fixed to 1 bits of CR4.
L4_VM_VMX_VMCS_ENUM_REG	VMCS enumeration info.
L4_VM_VMX_PROCBASED_CTLN2_REG	Processor based control 2 caps.
L4_VM_VMX_EPT_VPID_CAP_REG	EPT and VPID caps.
L4_VM_VMX_NUM_CAPS_REGS	Total number of VMX capability registers.

Definition at line 39 of file [__vm-vmx.h](#).

12.111.2.3 L4_vm_vmx_dfl1_regs

enum [L4_vm_vmx_dfl1_regs](#)

Exported VMX capability registers (default to 1 bits).

Enumerator

L4_VM_VMX_PINBASED_CTL5_DFL1_REG	Default 1 bits in pin-based controls.
L4_VM_VMX_PROCBASED_CTL5_DFL1_REG	Default 1 bits in processor-based controls.
L4_VM_VMX_EXIT_CTL5_DFL1_REG	Default 1 bits in exit controls.
L4_VM_VMX_ENTRY_CTL5_DFL1_REG	Default 1 bits in entry controls.
L4_VM_VMX_NUM_DFL1_REGS	Total number of default on registers.

Definition at line 62 of file [__vm-vmx.h](#).

12.111.3 Function Documentation

12.111.3.1 l4_vm_vmx_clear()

```
void l4_vm_vmx_clear (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Saves cached state from the kernel VMCS to the user VMCS.

Parameters

<i>vmcs</i>	Pointer to the kernel VMCS.
<i>user_vmcs</i>	Pointer to the user VMCS.

This function is comparable to VMX vmclear.

Definition at line 433 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_ptr_load\(\)](#).

Here is the caller graph for this function:



12.111.3.2 `l4_vm_vmx_field_len()`

```
unsigned l4_vm_vmx_field_len (
    unsigned field ) [inline]
```

Return length in bytes of a VMCS field.

Parameters

<i>field</i>	Field number.
--------------	---------------

Returns

Width of field in bytes.

Definition at line 357 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#), and [l4_vm_vmx_field_order\(\)](#).

Here is the call graph for this function:

12.111.3.3 `l4_vm_vmx_field_order()`

```
unsigned l4_vm_vmx_field_order (
    unsigned field ) [inline]
```

Return length in power of two (bytes) of a VMCS field.

Parameters

<i>field</i>	Field number.
--------------	---------------

Returns

Width of field in power of two (bytes).

Definition at line 342 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_field_len\(\)](#).

Here is the caller graph for this function:



12.111.3.4 `l4_vm_vmx_get_caps()`

```
l4_uint64_t l4_vm_vmx_get_caps (
    void const * vcpu_state,
    unsigned cap_msr ) [inline]
```

Get a capability register for VMX.

Parameters

<i>vcpu_state</i>	Pointer to the VCPU state of the VCPU.
<i>cap_msr</i>	Caps register index (see L4_vm_vmx_caps_regs).

Returns

The value of the capability register.

Definition at line 529 of file `__vm-vmx.h`.

References [L4_VCPU_OFFSET_EXT_INFOS](#).

12.111.3.5 `l4_vm_vmx_get_caps_default1()`

```
l4_uint32_t l4_vm_vmx_get_caps_default1 (
    void const * vcpu_state,
    unsigned cap_msr ) [inline]
```

Get a default to one capability register for VMX.

Parameters

<i>vcpu_state</i>	Pointer to the VCPU state of the VCPU.
<i>cap_msr</i>	Default 1 caps register index (see L4_vm_vmx_dfl1_regs).

Returns

The value of the capability register.

Definition at line 537 of file [__vm-vmx.h](#).

References [L4_VCPU_OFFSET_EXT_INFOS](#), [L4_VM_VMX_NUM_CAPS_REGS](#), and [L4_VM_VMX_PINBASE_0_D_CTL5_DFL1_REG](#).

12.111.3.6 l4_vm_vmx_get_cr2_index()

```
l4_uint32_t l4_vm_vmx_get_cr2_index (
    void const * vmcs ) [inline]
```

Get the VMCS field index of the virtual CR2 register.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
-------------	-------------------------------

Returns

The field index used for the virtual CR2 register as used by the current Fiasco.OC interface.

The CR2 register is actually not in the hardware VMCS, however our VMMs run in user mode and need to have access to this register so we put it into our software version of the VMCS.

See also

[L4_VM_VMX_VMCS_CR2](#)

Definition at line 545 of file [__vm-vmx.h](#).

12.111.3.7 l4_vm_vmx_ptr_load()

```
void l4_vm_vmx_ptr_load (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Loads the *user_vmcs* as the current VMCS.

Parameters

<i>vmcs</i>	Pointer to the kernel VMCS.
<i>user_vmcs</i>	Pointer to the user VMCS.

This function is comparable to VMX vmprld.

Definition at line 445 of file [__vm-vmx.h](#).

References [l4_vm_vmx_clear\(\)](#).

Here is the call graph for this function:



12.111.3.8 l4_vm_vmx_read()

```
l4_uint64_t l4_vm_vmx_read (
    void * vmcs,
    unsigned field ) [inline]
```

Read any VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

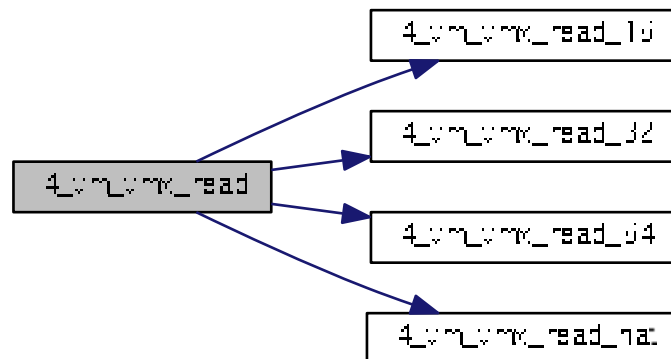
Returns

The value of the VMCS field with the given index.

Definition at line 481 of file [__vm-vmx.h](#).

References [l4_vm_vmx_read_16\(\)](#), [l4_vm_vmx_read_32\(\)](#), [l4_vm_vmx_read_64\(\)](#), and [l4_vm_vmx_read_nat\(\)](#).

Here is the call graph for this function:



12.111.3.9 l4_vm_vmx_read_16()

```

l4_uint16_t l4_vm_vmx_read_16 (
    void * vmcs,
    unsigned field ) [inline]
  
```

Read a 16bit VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

Returns

The value of the VMCS field with the given index.

Definition at line 466 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



12.111.3.10 `l4_vm_vmx_read_32()`

```
l4_uint32_t l4_vm_vmx_read_32 (  
    void * vmcs,  
    unsigned field ) [inline]
```

Read a 32bit VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

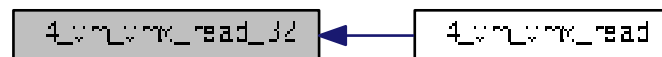
Returns

The value of the VMCS field with the given index.

Definition at line 471 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



12.111.3.11 `l4_vm_vmx_read_64()`

```
l4_uint64_t l4_vm_vmx_read_64 (  
    void * vmcs,  
    unsigned field ) [inline]
```

Read a 64bit VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

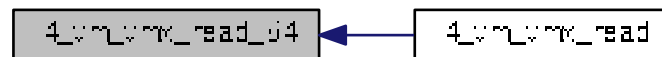
Returns

The value of the VMCS field with the given index.

Definition at line 476 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:

**12.111.3.12 l4_vm_vmx_read_nat()**

```
l4_umword_t l4_vm_vmx_read_nat (
    void * vmcs,
    unsigned field ) [inline]
```

Read a natural width VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

Returns

The value of the VMCS field with the given index.

Definition at line 461 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



12.111.3.13 `l4_vm_vmx_write()`

```
void l4_vm_vmx_write (
    void * vmcs,
    unsigned field,
    l4_uint64_t val ) [inline]
```

Write to an arbitrary VMCS field.

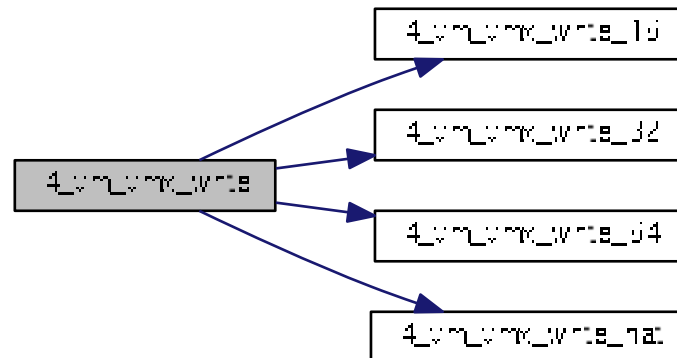
Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 516 of file `__vm-vmx.h`.

References `l4_vm_vmx_write_16()`, `l4_vm_vmx_write_32()`, `l4_vm_vmx_write_64()`, and `l4_vm_vmx_write_nat()`.

Here is the call graph for this function:

12.111.3.14 `l4_vm_vmx_write_16()`

```
void l4_vm_vmx_write_16 (
    void * vmcs,
    unsigned field,
    l4_uint16_t val ) [inline]
```

Write to a 16bit VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 500 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:

12.111.3.15 `l4_vm_vmx_write_32()`

```
void l4_vm_vmx_write_32 (
    void * vmcs,
    unsigned field,
    l4_uint32_t val ) [inline]
```

Write to a 32bit VMCS field.

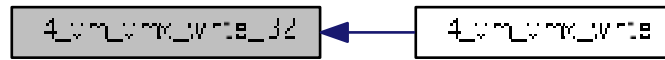
Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 505 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:



12.111.3.16 l4_vm_vmx_write_64()

```

void l4_vm_vmx_write_64 (
    void * vmcs,
    unsigned field,
    l4_uint64_t val ) [inline]
  
```

Write to a 64bit VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 510 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:



12.111.3.17 l4_vm_vmx_write_nat()

```

void l4_vm_vmx_write_nat (
    void * vmcs,
  
```

```
    unsigned field,  
    l4_umword_t val ) [inline]
```

Write to a natural width VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 495 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

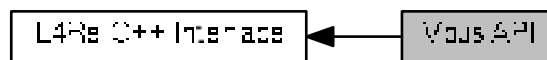
Here is the caller graph for this function:



12.112 Vbus API

C++ interface of the Vbus API.

Collaboration diagram for Vbus API:



Data Structures

- class [L4vbus::Pm< DEC >](#)
Power-management API mixin.
- class [L4vbus::Device](#)
Device on a [L4vbus::Vbus](#).
- class [L4vbus::Icu](#)
[Vbus](#) Interrupt controller API.
- class [L4vbus::Vbus](#)
The virtual bus ([Vbus](#)) interface.

12.112.1 Detailed Description

C++ interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Include File

```
#include <l4/vbus/vbus>
```

Refer to [L4 Vbus functions](#) for the C API.

12.113 Video API

Collaboration diagram for Video API:



Data Structures

- struct [l4re_video_color_component_t](#)
Color component structure.
- struct [l4re_video_pixel_info_t](#)
Pixel_info structure.
- struct [l4re_video_goos_info_t](#)
Goos information structure.
- struct [l4re_video_view_info_t](#)
View information structure.
- struct [l4re_video_view_t](#)
C representation of a goos view.

Typedefs

- typedef struct [l4re_video_color_component_t](#) [l4re_video_color_component_t](#)
Color component structure.
- typedef struct [l4re_video_pixel_info_t](#) [l4re_video_pixel_info_t](#)
Pixel_info structure.
- typedef struct [l4re_video_view_info_t](#) [l4re_video_view_info_t](#)
View information structure.
- typedef struct [l4re_video_view_t](#) [l4re_video_view_t](#)
C representation of a goos view.

Enumerations

- enum [l4re_video_goos_info_flags_t](#) { [F_l4re_video_goos_auto_refresh](#) = 0x01, [F_l4re_video_goos_pointer](#) = 0x02, [F_l4re_video_goos_dynamic_views](#) = 0x04, [F_l4re_video_goos_dynamic_buffers](#) = 0x08 }
- enum [l4re_video_view_info_flags_t](#) {
[F_l4re_video_view_none](#) = 0x00, [F_l4re_video_view_set_buffer](#) = 0x01, [F_l4re_video_view_set_buffer_offset](#) = 0x02, [F_l4re_video_view_set_bytes_per_line](#) = 0x04,
[F_l4re_video_view_set_pixel](#) = 0x08, [F_l4re_video_view_set_position](#) = 0x10, [F_l4re_video_view_dyn_allocated](#) = 0x20, [F_l4re_video_view_set_background](#) = 0x40,
[F_l4re_video_view_set_flags](#) = 0x80, [F_l4re_video_view_above](#) = 0x01000, [F_l4re_video_view_flags_mask](#) = 0xff000 }
Flags of information on a view.

Functions

- `int l4re_video_goos_info (l4re_video_goos_t goos, l4re_video_goos_info_t *ginfo) L4_NOTHROW`
Get information on a goos.
- `int l4re_video_goos_refresh (l4re_video_goos_t goos, int x, int y, int w, int h) L4_NOTHROW`
Flush a rectangle of pixels of the goos screen.
- `int l4re_video_goos_create_buffer (l4re_video_goos_t goos, unsigned long size, l4_cap_idx_t buffer) L4_NOTHROW`
Create a new buffer (memory buffer) for pixel data.
- `int l4re_video_goos_delete_buffer (l4re_video_goos_t goos, unsigned idx) L4_NOTHROW`
Delete a pixel buffer.
- `int l4re_video_goos_get_static_buffer (l4re_video_goos_t goos, unsigned idx, l4_cap_idx_t buffer) L4_NOTHROW`
Get the data-space capability of the static pixel buffer.
- `int l4re_video_goos_create_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`
Create a new view (.
- `int l4re_video_goos_delete_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`
Delete a view.
- `int l4re_video_goos_get_view (l4re_video_goos_t goos, unsigned idx, l4re_video_view_t *view) L4_NOTHROW`
Get a view for the given index.
- `int l4re_video_view_refresh (l4re_video_view_t *view, int x, int y, int w, int h) L4_NOTHROW`
Flush the given rectangle of pixels of the given view.
- `int l4re_video_view_get_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`
Retrieve information about the given view.
- `int l4re_video_view_set_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`
Set properties of the view.
- `int l4re_video_view_set_viewport (l4re_video_view_t *view, int x, int y, int w, int h, unsigned long bofs) L4_NOTHROW`
Set the viewport parameters of a view.
- `int l4re_video_view_stack (l4re_video_view_t *view, l4re_video_view_t *pivot, int behind) L4_NOTHROW`
Change the stacking order in the stack of visible views.

12.113.1 Detailed Description

12.113.2 Typedef Documentation

12.113.2.1 l4re_video_view_t

```
typedef struct l4re_video_view_t l4re_video_view_t
```

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

12.113.3 Enumeration Type Documentation

12.113.3.1 l4re_video_goos_info_flags_t

enum [l4re_video_goos_info_flags_t](#)

Flags of information on the goos.

Enumerator

F_l4re_video_goos_auto_refresh	The graphics display is automatically refreshed.
F_l4re_video_goos_pointer	We have a mouse pointer.
F_l4re_video_goos_dynamic_views	Supports dynamically allocated views.
F_l4re_video_goos_dynamic_buffers	Supports dynamically allocated buffers.

Definition at line 39 of file [goos.h](#).

12.113.3.2 l4re_video_view_info_flags_t

enum [l4re_video_view_info_flags_t](#)

Flags of information on a view.

Enumerator

F_l4re_video_view_none	everything for this view is static (the VESA-FB case)
F_l4re_video_view_set_buffer	buffer object for this view can be changed
F_l4re_video_view_set_buffer_offset	buffer offset can be set
F_l4re_video_view_set_bytes_per_line	bytes per line can be set
F_l4re_video_view_set_pixel	pixel type can be set
F_l4re_video_view_set_position	position on screen can be set
F_l4re_video_view_dyn_allocated	View is dynamically allocated.
F_l4re_video_view_set_background	Set view as background for session.
F_l4re_video_view_set_flags	Set view property flags.
F_l4re_video_view_above	Flag the view as stay on top.
F_l4re_video_view_flags_mask	Mask containing all possible property flags.

Definition at line 33 of file [view.h](#).

12.113.4 Function Documentation

12.113.4.1 l4re_video_goos_create_buffer()

```
int l4re_video_goos_create_buffer (
    l4re_video_goos_t goos,
    unsigned long size,
    l4_cap_idx_t buffer )
```

Create a new buffer (memory buffer) for pixel data.

Parameters

<i>goos</i>	the target object for the operation.
<i>size</i>	the size in bytes for the pixel buffer.
<i>buffer</i>	a capability index to receive the data-space capability for the buffer.

Returns

≥ 0 : The index of the created buffer (used to assign views and for deletion). < 0 : on error

12.113.4.2 l4re_video_goos_create_view()

```
int l4re_video_goos_create_view (
    l4re_video_goos_t goos,
    l4re_video_view_t * view )
```

Create a new view (.

See also

[l4re_video_view_t](#)

Parameters

<i>goos</i>	the goos session to use.
-------------	--------------------------

Return values

<i>view</i>	the structure will be initialized for the new view.
-------------	---

12.113.4.3 l4re_video_goos_delete_buffer()

```
int l4re_video_goos_delete_buffer (
    l4re_video_goos_t goos,
    unsigned idx )
```

Delete a pixel buffer.

Parameters

<i>goos</i>	the target goos object.
<i>idx</i>	the buffer index of the buffer to delete (the return value of l4re_video_goos_create_buffer())

12.113.4.4 `l4re_video_goos_delete_view()`

```
int l4re_video_goos_delete_view (
    l4re_video_goos_t goos,
    l4re_video_view_t * view )
```

Delete a view.

Parameters

<i>goos</i>	the goos session to use.
<i>view</i>	the view to delete, the given data-structure is invalid afterwards.

12.113.4.5 `l4re_video_goos_get_static_buffer()`

```
int l4re_video_goos_get_static_buffer (
    l4re_video_goos_t goos,
    unsigned idx,
    l4_cap_idx_t buffer )
```

Get the data-space capability of the static pixel buffer.

Parameters

<i>goos</i>	The target goos object.
<i>idx</i>	Index of the static buffer.
<i>buffer</i>	A capability index to receive the data-space capability.

This function allows access to static, preexisting pixel buffers. Such static buffers exist for static configurations, such as the VESA framebuffer.

12.113.4.6 `l4re_video_goos_get_view()`

```
int l4re_video_goos_get_view (
    l4re_video_goos_t goos,
    unsigned idx,
    l4re_video_view_t * view )
```

Get a view for the given index.

Parameters

<i>goos</i>	the target goos session.
<i>idx</i>	the index of the view to retrieve.

Return values

<i>view</i>	the structure will be initialized to the view with the given index.
-------------	---

This function allows to access static views as provided by the VESA framebuffer (the monitor). However, it also allows to access dynamic views created with [l4re_video_goos_create_view\(\)](#).

12.113.4.7 l4re_video_goos_info()

```
int l4re_video_goos_info (
    l4re_video_goos_t goos,
    l4re_video_goos_info_t * ginfo )
```

Get information on a goos.

Parameters

<i>goos</i>	Goos object
-------------	-------------

Return values

<i>ginfo</i>	Pointer to goos information structure.
--------------	--

Returns

0 for success, <0 on error

- [-L4_ENODEV](#)
- IPC errors

12.113.4.8 l4re_video_goos_refresh()

```
int l4re_video_goos_refresh (
    l4re_video_goos_t goos,
    int x,
    int y,
    int w,
    int h )
```

Flush a rectangle of pixels of the goos screen.

Parameters

<i>goos</i>	the target object of the operation.
<i>x</i>	the x-coordinate of the upper left corner of the rectangle
<i>y</i>	the y-coordinate of the upper left corner of the rectangle
<i>w</i>	the width of the rectangle to be flushed
<i>h</i>	the height of the rectangle

12.113.4.9 l4re_video_view_get_info()

```
int l4re_video_view_get_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info )
```

Retrieve information about the given *view*.

Parameters

<i>view</i>	the target view for the operation.
-------------	------------------------------------

Return values

<i>info</i>	a buffer receiving the information about the view.
-------------	--

12.113.4.10 l4re_video_view_refresh()

```
int l4re_video_view_refresh (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h )
```

Flush the given rectangle of pixels of the given *view*.

Parameters

<i>view</i>	the target view of the operation.
<i>x</i>	x-coordinate of the upper left corner
<i>y</i>	y-coordinate of the upper left corner
<i>w</i>	the width of the rectangle
<i>h</i>	the height of the rectangle

12.113.4.11 `l4re_video_view_set_info()`

```
int l4re_video_view_set_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info )
```

Set properties of the view.

Parameters

<i>view</i>	the target view of the operation.
<i>info</i>	the parameters to be set on the view.

Which parameters can be manipulated on a given view can be figured out with `l4re_video_view_get_info()` and this depends on the concrete instance the view object.

12.113.4.12 `l4re_video_view_set_viewport()`

```
int l4re_video_view_set_viewport (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h,
    unsigned long bofs )
```

Set the viewport parameters of a view.

Parameters

<i>view</i>	the target view of the operation.
<i>x</i>	the x-coordinate of the upper left corner on the screen.
<i>y</i>	the y-coordinate of the upper left corner on the screen.
<i>w</i>	the width of the view.
<i>h</i>	the height of the view.
<i>bofs</i>	the offset (in bytes) of the upper left pixel in the memory buffer

This function is a convenience wrapper for `l4re_video_view_set_info()`, just setting the often changed parameters of a dynamic view. With this function a view can be placed on the real screen and at the same time on its backing buffer.

12.113.4.13 `l4re_video_view_stack()`

```
int l4re_video_view_stack (
    l4re_video_view_t * view,
    l4re_video_view_t * pivot,
    int behind )
```

Change the stacking order in the stack of visible views.

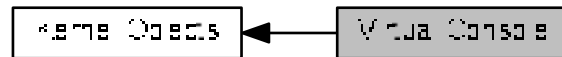
Parameters

<i>view</i>	the target view for the operation.
<i>pivot</i>	the neighbor view, relative to which <i>view</i> shall be stacked. a NULL value allows top (<i>behind</i> = 1) and bottom (<i>behind</i> = 0) placement of the view.
<i>behind</i>	describes the placement of the view relative to the <i>pivot</i> view.

12.114 Virtual Console

Virtual console for simple character based input and output.

Collaboration diagram for Virtual Console:



Data Structures

- struct [l4_vcon_attr_t](#)
Vcon attribute structure.

Typedefs

- typedef struct [l4_vcon_attr_t](#) [l4_vcon_attr_t](#)
Vcon attribute structure.

Enumerations

- enum [L4_vcon_size_consts](#) { [L4_VCON_WRITE_SIZE](#) = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t), [L4_VCON_READ_SIZE](#) = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t) }
Size constants.
- enum [L4_vcon_i_flags](#) { [L4_VCON_INLCR](#) = 000100, [L4_VCON_IGNCR](#) = 000200, [L4_VCON_ICRNL](#) = 000400 }
Input flags.
- enum [L4_vcon_o_flags](#) { [L4_VCON_ONLCR](#) = 000004, [L4_VCON_OCRNL](#) = 000010, [L4_VCON_ONLRET](#) = 000040 }
Output flags.
- enum [L4_vcon_l_flags](#) { [L4_VCON_ICANON](#) = 000002, [L4_VCON_ECHO](#) = 000010 }
Local flags.

Functions

- [l4_msgtag_t l4_vcon_send](#) ([l4_cap_idx_t](#) vcon, char const *buf, unsigned size) [L4_NOTHROW](#)
Send data to virtual console.
- [l4_msgtag_t l4_vcon_send_u](#) ([l4_cap_idx_t](#) vcon, char const *buf, unsigned size, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Send data to *this* virtual console.
- long [l4_vcon_write](#) ([l4_cap_idx_t](#) vcon, char const *buf, unsigned size) [L4_NOTHROW](#)
Write data to virtual console.
- long [l4_vcon_write_u](#) ([l4_cap_idx_t](#) vcon, char const *buf, unsigned size, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Write data to *this* virtual console.
- int [l4_vcon_read](#) ([l4_cap_idx_t](#) vcon, char *buf, unsigned size) [L4_NOTHROW](#)
Read data from virtual console.
- int [l4_vcon_read_u](#) ([l4_cap_idx_t](#) vcon, char *buf, unsigned size, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Read data from *this* virtual console.
- int [l4_vcon_read_with_flags](#) ([l4_cap_idx_t](#) vcon, char *buf, unsigned size) [L4_NOTHROW](#)
Read data from virtual console, extended version including flags.
- [l4_msgtag_t l4_vcon_set_attr](#) ([l4_cap_idx_t](#) vcon, [l4_vcon_attr_t](#) const *attr) [L4_NOTHROW](#)
Set attributes of a Vcon.
- [l4_msgtag_t l4_vcon_set_attr_u](#) ([l4_cap_idx_t](#) vcon, [l4_vcon_attr_t](#) const *attr, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Set the attributes of *this* virtual console.
- [l4_msgtag_t l4_vcon_get_attr](#) ([l4_cap_idx_t](#) vcon, [l4_vcon_attr_t](#) *attr) [L4_NOTHROW](#)
Get attributes of a Vcon.
- [l4_msgtag_t l4_vcon_get_attr_u](#) ([l4_cap_idx_t](#) vcon, [l4_vcon_attr_t](#) *attr, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Get attributes of *this* virtual console.

12.114.1 Detailed Description

Virtual console for simple character based input and output.

The interrupt for read events is provided by the virtual key interrupt.

Include File

```
#include <l4/sys/vcon.h>
```

See [L4::Vcon](#) for the C++ interface.

12.114.2 Enumeration Type Documentation

12.114.2.1 L4_vcon_i_flags

```
enum L4_vcon_i_flags
```

Input flags.

Enumerator

L4_VCON_INLCR	Translate NL to CR.
L4_VCON_IGNCR	Ignore CR.
L4_VCON_ICRNL	Translate CR to NL if L4_VCON_IGNCR is not set.

Definition at line 189 of file [vcon.h](#).

12.114.2.2 L4_vcon_l_flags

enum [L4_vcon_l_flags](#)

Local flags.

Enumerator

L4_VCON_ICANON	Canonical mode.
L4_VCON_ECHO	Echo input.

Definition at line 211 of file [vcon.h](#).

12.114.2.3 L4_vcon_o_flags

enum [L4_vcon_o_flags](#)

Output flags.

Enumerator

L4_VCON_ONLCR	Translate NL to CR-NL.
L4_VCON_OCRNL	Translate CR to NL.
L4_VCON_ONLRET	Do not output CR.

Definition at line 200 of file [vcon.h](#).

12.114.2.4 L4_vcon_size_consts

enum [L4_vcon_size_consts](#)

Size constants.

Enumerator

L4_VCON_WRITE_SIZE	Maximum size that can be written with one l4_vcon_write call.
L4_VCON_READ_SIZE	Maximum size that can be read with one l4_vcon_read* call.

Definition at line 95 of file [vcon.h](#).

12.114.3 Function Documentation

12.114.3.1 l4_vcon_get_attr()

```
l4_msgtag_t l4_vcon_get_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr ) [inline]
```

Get attributes of a Vcon.

Parameters

	<i>vcon</i>	Vcon object.
out	<i>attr</i>	Attribute structure.

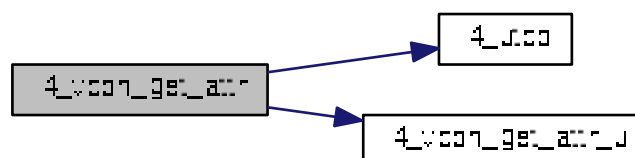
Returns

Syscall return tag

Definition at line 409 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



12.114.3.2 l4_vcon_get_attr_u()

```
l4_msgtag_t l4_vcon_get_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr,
    l4_utcb_t * utcb ) [inline]
```

Get attributes of this virtual console.

Parameters

	<i>vcon</i>	Capability index of the vcon object.
out	<i>attr</i>	Attribute structure. Contains the attributes after a successful call of this function.
	<i>utcb</i>	UTCB pointer of the calling thread.

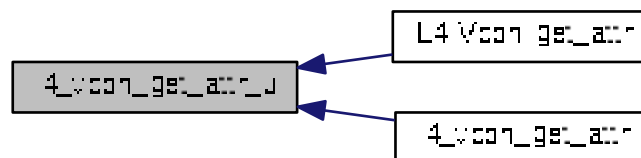
Returns

Syscall return tag.

Definition at line 391 of file [vcon.h](#).

Referenced by [L4::Vcon::get_attr\(\)](#), and [l4_vcon_get_attr\(\)](#).

Here is the caller graph for this function:



12.114.3.3 l4_vcon_read()

```
int l4_vcon_read (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size ) [inline]
```

Read data from virtual console.

Parameters

	<i>vcon</i>	Vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of buffer in bytes.

Return values

<code><0</code>	Error code.
<code>>size</code>	If more bytes are to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code><=size</code>	Number of bytes read.

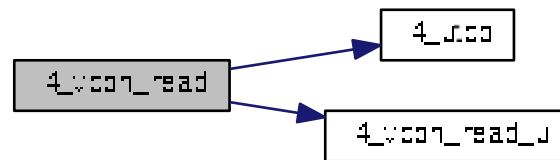
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line [365](#) of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:

12.114.3.4 `l4_vcon_read_u()`

```

int l4_vcon_read_u (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size,
    l4_utcb_t * utcb ) [inline]

```

Read data from this virtual console.

Parameters

	<i>vcon</i>	Capability index of the vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB pointer of the calling thread.

Return values

<code><0</code>	Error code.
--------------------	-------------

Return values

$>size$	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
$\leq size$	Number of bytes read.

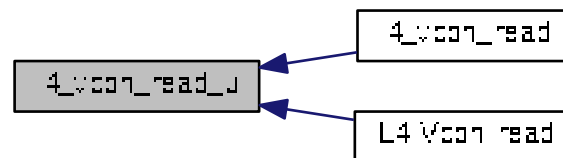
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line [355](#) of file [vcon.h](#).

Referenced by [l4_vcon_read\(\)](#), and [L4::Vcon::read\(\)](#).

Here is the caller graph for this function:



12.114.3.5 l4_vcon_read_with_flags()

```

int l4_vcon_read_with_flags (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size ) [inline]
  
```

Read data from virtual console, extended version including flags.

Parameters

	<i>vcon</i>	Vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of buffer in bytes.

If this function returns a positive value the caller can check the [L4_VCON_READ_STAT_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4_VCON_READ_SIZE_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

buf might be a NULL, in this case the input data will be dropped.

Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Return values

<code>< 0</code>	Error code.
<code>> size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code><=size</code>	Number of bytes read.

Definition at line [349](#) of file [vcon.h](#).

12.114.3.6 l4_vcon_send()

```
l4_msgtag_t l4_vcon_send (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size ) [inline]
```

Send data to virtual console.

Parameters

<code>vcon</code>	Vcon object.
<code>buf</code>	Pointer to data buffer.
<code>size</code>	Size of buffer in bytes.

Returns

Syscall return tag

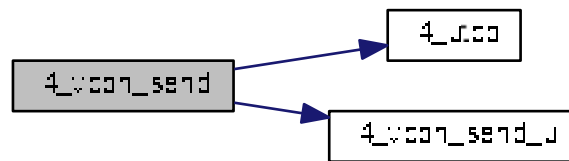
Note

Size must not exceed [L4_VCON_WRITE_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for send errors, do not use [l4_error\(\)](#), as [l4_error\(\)](#) will always return an error.

Definition at line [289](#) of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:



12.114.3.7 l4_vcon_send_u()

```

l4_msgtag_t l4_vcon_send_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb ) [inline]
  
```

Send data to this virtual console.

Parameters

<i>vcon</i>	Capability index of the Vcon object.
<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTBC pointer of the calling thread.

Returns

Syscall return tag

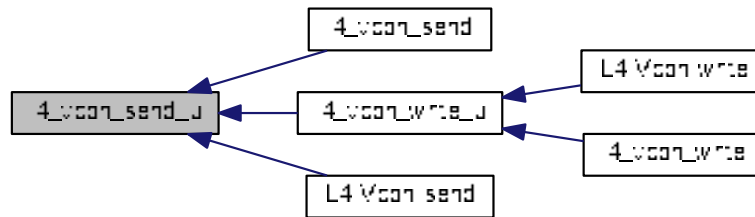
Note

Size must not exceed [L4_VCON_WRITE_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for send errors, do not use [l4_error\(\)](#), as [l4_error\(\)](#) will always return an error.

Definition at line 275 of file [vcon.h](#).

Referenced by [l4_vcon_send\(\)](#), [l4_vcon_write_u\(\)](#), and [L4::Vcon::send\(\)](#).

Here is the caller graph for this function:



12.114.3.8 l4_vcon_set_attr()

```
l4_msgtag_t l4_vcon_set_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr ) [inline]
```

Set attributes of a Vcon.

Parameters

<i>vcon</i>	Vcon object.
<i>attr</i>	Attribute structure.

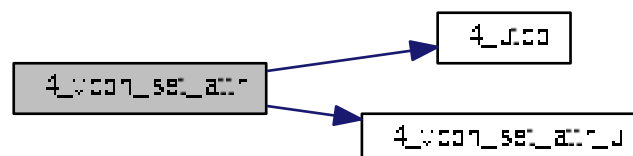
Returns

Syscall return tag

Definition at line 385 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



12.114.3.9 l4_vcon_set_attr_u()

```
l4_msgtag_t l4_vcon_set_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb ) [inline]
```

Set the attributes of this virtual console.

Parameters

<i>vcon</i>	Capability index of the vcon object.
<i>attr</i>	Attribute structure with the attributes for the virtual console.
<i>utcb</i>	UTCB pointer of the calling thread.

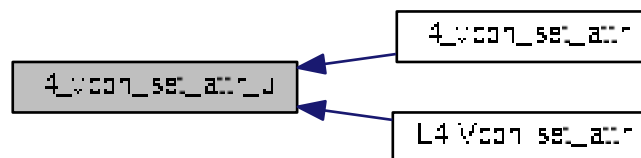
Returns

Syscall return tag.

Definition at line 371 of file [vcon.h](#).

Referenced by [l4_vcon_set_attr\(\)](#), and [L4::Vcon::set_attr\(\)](#).

Here is the caller graph for this function:



12.114.3.10 l4_vcon_write()

```
long l4_vcon_write (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size ) [inline]
```

Write data to virtual console.

Parameters

<i>vcon</i>	Vcon object.
<i>buf</i>	Pointer to data buffer.
<i>size</i>	Size of buffer in bytes.

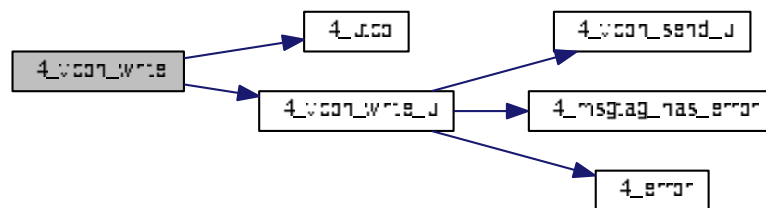
Return values

< 0	Error.
≥ 0	Number of bytes written to the virtual console

Definition at line 310 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



12.114.3.11 l4_vcon_write_u()

```

long l4_vcon_write_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb ) [inline]

```

Write data to this virtual console.

Parameters

<i>vcon</i>	Capability index of the vcon object.
<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB pointer of the calling thread.

Return values

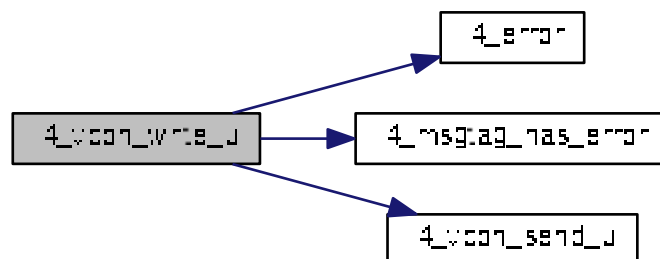
< 0	Error.
≥ 0	Number of bytes written to the virtual console.

Definition at line 295 of file [vcon.h](#).

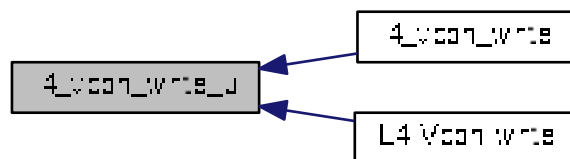
References [l4_error\(\)](#), [l4_msgtag_has_error\(\)](#), [l4_vcon_send_u\(\)](#), and [L4_VCON_WRITE_SIZE](#).

Referenced by [l4_vcon_write\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the call graph for this function:



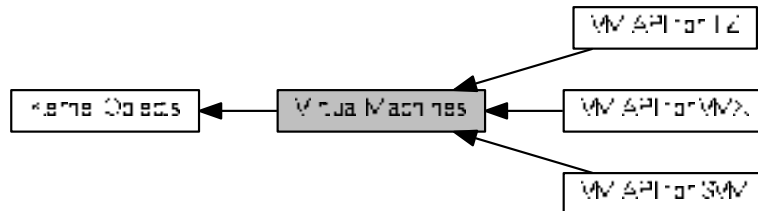
Here is the caller graph for this function:



12.115 Virtual Machines

Virtual Machine API.

Collaboration diagram for Virtual Machines:



Modules

- [VM API for SVM](#)
Virtual machine API for SVM.
- [VM API for TZ](#)
Virtual Machine API for ARM TrustZone.
- [VM API for VMX](#)
Virtual machine API for VMX.

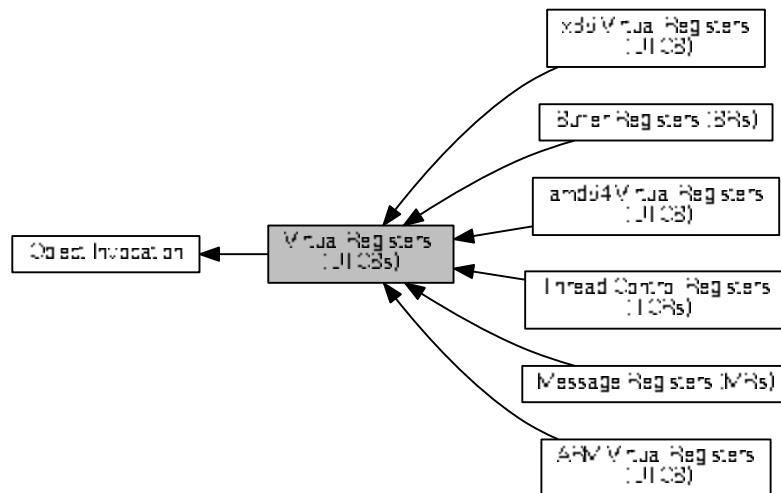
12.115.1 Detailed Description

Virtual Machine API.

12.116 Virtual Registers (UTCBs)

[L4](#) Virtual Registers (UTCB).

Collaboration diagram for Virtual Registers (UTCBs):



Modules

- [ARM Virtual Registers \(UTCB\)](#)
- [Buffer Registers \(BRs\)](#)
- [Message Registers \(MRs\)](#)
- [Thread Control Registers \(TCRs\)](#)
- [amd64 Virtual Registers \(UTCB\)](#)
- [x86 Virtual Registers \(UTCB\)](#)

Files

- file [utcb.h](#)
UTCB definitions for ARM.
- file [utcb.h](#)
UTCB definitions for amd64.
- file [utcb.h](#)
UTCB definitions for X86.

Typedefs

- typedef struct [l4_utcb_t](#) [l4_utcb_t](#)
Opaque type for the UTCB.

Functions

- [l4_utcb_t * l4_utcb](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the UTCB address.
- [l4_msg_regs_t * l4_utcb_mr](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the message-register block of a UTCB.
- [l4_buf_regs_t * l4_utcb_br](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the buffer-register block of a UTCB.
- [l4_thread_regs_t * l4_utcb_tcr](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the thread-control-register block of a UTCB.

12.116.1 Detailed Description

[L4](#) Virtual Registers (UTCB).

Include File

```
#include <l4/sys/utcb.h>
```

The virtual registers are part of the micro-kernel API and are located in the user-level thread control block (UTCB). The UTCB is a data structure defined by the micro kernel and located on kernel-provided memory. Each [L4](#) thread gets a unique UTCB assigned when it is bound to a task (see [Thread Control](#) , [l4_thread_control_bind\(\)](#) for more information).

The UTCB is arranged in three blocks of virtual registers.

- [Thread Control Registers \(TCRs\)](#)
- [Message Registers \(MRs\)](#)
- [Buffer Registers \(BRs\)](#)

To access the contents of the virtual registers the [l4_utcb_mr\(\)](#), [l4_utcb_tcr\(\)](#), and [l4_utcb_br\(\)](#) functions must be used.

12.116.2 Typedef Documentation

12.116.2.1 [l4_utcb_t](#)

```
typedef struct l4\_utcb\_t l4\_utcb\_t
```

Opaque type for the UTCB.

To access the contents of the virtual registers the [l4_utcb_mr\(\)](#), [l4_utcb_tcr\(\)](#), and [l4_utcb_br\(\)](#) functions must be used.

Definition at line 67 of file [utcb.h](#).

12.116.3 Function Documentation

12.116.3.1 l4_utcb_br()

```
l4_buf_regs_t * l4_utcb_br (
    void ) [inline]
```

Get the buffer-register block of a UTCB.

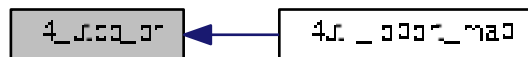
Returns

A pointer to the buffer-register block of u.

Definition at line 355 of file [utcb.h](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



12.116.3.2 l4_utcb_mr()

```
l4_msg_regs_t * l4_utcb_mr (
    void ) [inline]
```

Get the message-register block of a UTCB.

Returns

A pointer to the message-register block of u.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 352 of file [utcb.h](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



12.116.3.3 l4_tcb_tcr()

```
l4_thread_regs_t * l4_tcb_tcr (
    void ) [inline]
```

Get the thread-control-register block of a UTCB.

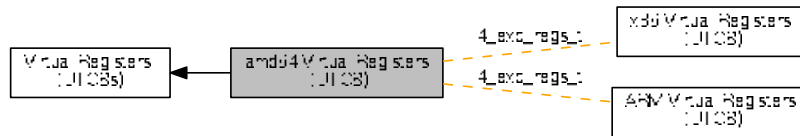
Returns

A pointer to the thread-control-register block of `u`.

Definition at line 358 of file [utcb.h](#).

12.117 amd64 Virtual Registers (UTCB)

Collaboration diagram for amd64 Virtual Registers (UTCB):



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Typedefs

- typedef struct [l4_exc_regs_t](#) [l4_exc_regs_t](#)
UTCB structure for exceptions.

Enumerations

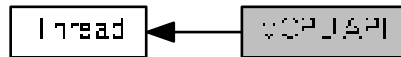
- enum [L4_utcb_consts_amd64](#)
UTCB constants for AMD64.

12.117.1 Detailed Description

12.118 vCPU API

vCPU API

Collaboration diagram for vCPU API:



Data Structures

- struct [l4_vcpu_state_t](#)
State of a vCPU.
- struct [l4_vcpu_regs_t](#)
vCPU registers.
- struct [l4_vcpu_ipc_regs_t](#)
vCPU message registers.

Typedefs

- typedef struct [l4_vcpu_state_t](#) [l4_vcpu_state_t](#)
State of a vCPU.
- typedef struct [l4_vcpu_regs_t](#) [l4_vcpu_regs_t](#)
vCPU registers.
- typedef struct [l4_vcpu_ipc_regs_t](#) [l4_vcpu_ipc_regs_t](#)
vCPU message registers.
- typedef struct [l4_vcpu_regs_t](#) [l4_vcpu_regs_t](#)
vCPU registers.
- typedef struct [l4_vcpu_ipc_regs_t](#) [l4_vcpu_ipc_regs_t](#)
vCPU message registers.
- typedef struct [l4_vcpu_regs_t](#) [l4_vcpu_regs_t](#)
vCPU registers.
- typedef struct [l4_vcpu_ipc_regs_t](#) [l4_vcpu_ipc_regs_t](#)
vCPU message registers.

Enumerations

- enum [L4_vcpu_state_flags](#) {
[L4_VCPU_F_IRQ](#) = 0x01, [L4_VCPU_F_PAGE_FAULTS](#) = 0x02, [L4_VCPU_F_EXCEPTIONS](#) = 0x04, [L4_VCPU_F_DEBUG_EXC](#) = 0x08,
[L4_VCPU_F_USER_MODE](#) = 0x20, [L4_VCPU_F_FPU_ENABLED](#) = 0x80 }
State flags of a vCPU.
- enum [L4_vcpu_sticky_flags](#) { [L4_VCPU_SF_IRQ_PENDING](#) = 0x01 }
Sticky flags of a vCPU.
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x400, [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
Offsets for vCPU state layouts.

12.118.1 Detailed Description

vCPU API

The vCPU API in [L4Re](#) implements virtual processors (vCPUs) on top of [L4::Thread](#). This API can be used for user level threading, operating system rehosting (see L4Linux) and virtualization.

You switch a thread into vCPU operation with [L4::Thread::vcpu_control](#).

Extended vCPU operation is used for hardware CPU virtualization. It can be enabled with [L4::Thread::vcpu_control_ext](#).

[vCPU Support Library](#) defines a convenience API for working with vCPUs.

See also

[vCPU Support Library](#)

12.118.2 Enumeration Type Documentation

12.118.2.1 L4_vcpu_state_flags

enum [L4_vcpu_state_flags](#)

State flags of a vCPU.

Enumerator

L4_VCPU_F_IRQ	IRQs (events) enabled.
L4_VCPU_F_PAGE_FAULTS	Page faults enabled.
L4_VCPU_F_EXCEPTIONS	Exception enabled.
L4_VCPU_F_DEBUG_EXC	Debug exception enabled.
L4_VCPU_F_USER_MODE	User task will be used.
L4_VCPU_F_FPU_ENABLED	FPU enabled.

Definition at line 71 of file [vcpu.h](#).

12.118.2.2 L4_vcpu_state_offset

enum [L4_vcpu_state_offset](#)

Offsets for vCPU state layouts.

Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 94 of file [vcpu.h](#).

12.118.2.3 L4_vcpu_sticky_flags

enum [L4_vcpu_sticky_flags](#)

Sticky flags of a vCPU.

Enumerator

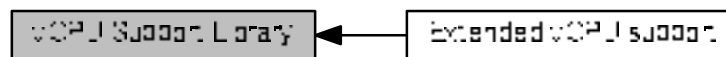
L4_VCPU_SF_IRQ_PENDING	An event (e.g. IRQ) is pending.
------------------------	---------------------------------

Definition at line 85 of file [vcpu.h](#).

12.119 vCPU Support Library

vCPU handling functionality.

Collaboration diagram for vCPU Support Library:



Modules

- [Extended vCPU support](#)
extended vCPU handling functionality.

Data Structures

- class [L4vcpu::State](#)
C++ implementation of state word in the vCPU area.
- class [L4vcpu::Vcpu](#)
C++ implementation of the vCPU save state area.

Functions

- void [l4vcpu_irq_disable](#) ([l4_vcpu_state_t](#) *vcpu) [L4_NOTHROW](#)
Disable a vCPU for event delivery.
- unsigned [l4vcpu_irq_disable_save](#) ([l4_vcpu_state_t](#) *vcpu) [L4_NOTHROW](#)
Disable a vCPU for event delivery and return previous state.
- void [l4vcpu_irq_enable](#) ([l4_vcpu_state_t](#) *vcpu, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)
Enable a vCPU for event delivery.
- void [l4vcpu_irq_restore](#) ([l4_vcpu_state_t](#) *vcpu, unsigned s, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)
Restore a previously saved IRQ/event state.
- void [l4vcpu_wait_for_event](#) ([l4_vcpu_state_t](#) *vcpu, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)
Wait for event.
- void [l4vcpu_print_state](#) (const [l4_vcpu_state_t](#) *vcpu, const char *prefix) [L4_NOTHROW](#)
Print the state of a vCPU.
- int [l4vcpu_is_irq_entry](#) ([l4_vcpu_state_t](#) const *vcpu) [L4_NOTHROW](#)
Return whether the entry reason was an IRQ/IPC message.
- int [l4vcpu_is_page_fault_entry](#) ([l4_vcpu_state_t](#) const *vcpu) [L4_NOTHROW](#)
Return whether the entry reason was a page fault.

12.119.1 Detailed Description

vCPU handling functionality.

This library provides convenience functionality on top of the l4sys vCPU interface to ease programming. It wraps commonly used code and abstracts architecture depends parts as far as reasonable.

12.119.2 Function Documentation

12.119.2.1 l4vcpu_irq_disable()

```
void l4vcpu_irq_disable (
    l4_vcpu_state_t * vcpu ) [inline]
```

Disable a vCPU for event delivery.

Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

Definition at line 208 of file [vcpu.h](#).

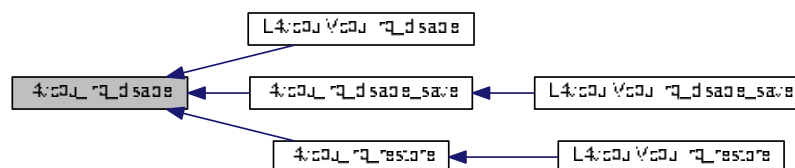
References [l4_barrier\(\)](#), [L4_CV](#), and [L4_VCPU_F_IRQ](#).

Referenced by [L4vcpu::Vcpu::irq_disable\(\)](#), [l4vcpu_irq_disable_save\(\)](#), and [l4vcpu_irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.119.2.2 l4vcpu_irq_disable_save()

```
unsigned l4vcpu_irq_disable_save (
    l4_vcpu_state_t * vcpu ) [inline]
```

Disable a vCPU for event delivery and return previous state.

Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

Returns

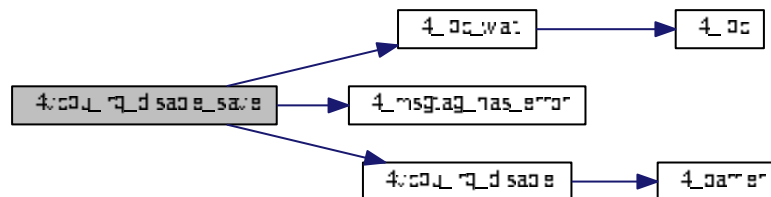
IRQ state before disabling IRQs.

Definition at line 216 of file [vcpu.h](#).

References [l4_vcpu_state_t::i](#), [L4_CV](#), [l4_ipc_wait\(\)](#), [L4_LIKELY](#), [l4_msgtag_has_error\(\)](#), [L4_NOTHROW](#), and [l4vcpu_irq_disable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq_disable_save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.119.2.3 l4vcpu_irq_enable()

```
void l4vcpu_irq_enable (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hdl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) [inline]
```

Enable a vCPU for event delivery.

Parameters

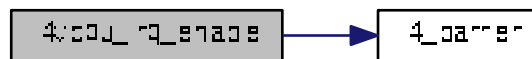
<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	Utc b pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 239 of file [vcpu.h](#).

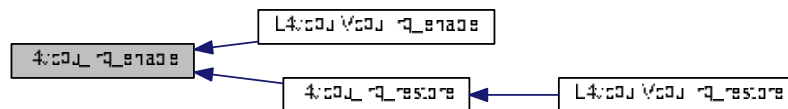
References [l4_barrier\(\)](#), [L4_CV](#), [L4_IPC_BOTH_TIMEOUT_0](#), [L4_LIKELY](#), [L4_VCPU_F_IRQ](#), [L4_VCPU_SF_IRQ_PENDING](#), [l4_vcpu_state_t::state](#), and [l4_vcpu_state_t::sticky_flags](#).

Referenced by [L4vcpu::Vcpu::irq_enable\(\)](#), and [l4vcpu_irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.119.2.4 l4vcpu_irq_restore()

```

void l4vcpu_irq_restore (
    l4_vcpu_state_t * vcpu,
    unsigned s,
    l4_utcb_t * utcb,
    l4vcpu_event_hdl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) [inline]
  
```

Restore a previously saved IRQ/event state.

Parameters

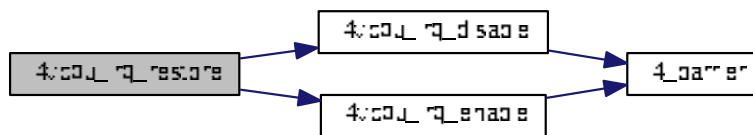
<i>vcpu</i>	Pointer to vCPU area.
<i>s</i>	IRQ state to be restored.
<i>utcb</i>	Utcbl pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending after enabling.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 264 of file [vcpu.h](#).

References [L4_CV](#), [L4_VCPU_F_IRQ](#), [l4vcpu_irq_disable\(\)](#), [l4vcpu_irq_enable\(\)](#), and [l4_vcpu_state_t::state](#).

Referenced by [L4vcpu::Vcpu::irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.119.2.5 l4vcpu_is_irq_entry()

```
int l4vcpu_is_irq_entry (
    l4\_vcpu\_state\_t const * vcpu ) [inline]
```

Return whether the entry reason was an IRQ/IPC message.

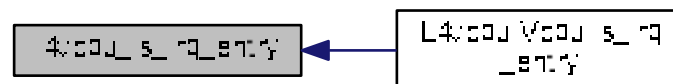
Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

return 0 if not, !=0 otherwise.

Referenced by [L4vcpu::Vcpu::is_irq_entry\(\)](#).

Here is the caller graph for this function:



12.119.2.6 l4vcpu_is_page_fault_entry()

```
int l4vcpu_is_page_fault_entry (
    l4_vcpu_state_t const * vcpu ) [inline]
```

Return whether the entry reason was a page fault.

Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

return 0 if not, !=0 otherwise.

Referenced by [L4vcpu::Vcpu::is_page_fault_entry\(\)](#).

Here is the caller graph for this function:



12.119.2.7 l4vcpu_print_state()

```
void l4vcpu_print_state (
    const l4_vcpu_state_t * vcpu,
    const char * prefix )
```

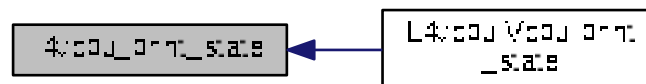
Print the state of a vCPU.

Parameters

<i>vcpu</i>	Pointer to vCPU area.
<i>prefix</i>	A prefix for each line printed.

Referenced by [L4vcpu::Vcpu::print_state\(\)](#).

Here is the caller graph for this function:



12.119.2.8 l4vcpu_wait_for_event()

```
void l4vcpu_wait_for_event (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc ) [inline]
```

Wait for event.

Parameters

<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	UtcB pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called when the vCPU awakes and needs to handle an event/IRQ.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation.

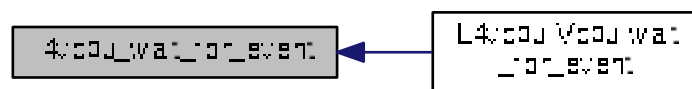
Note that event delivery remains disabled after this function returns.

Definition at line 277 of file [vcpu.h](#).

References [__END_DECLS](#), and [L4_IPC_NEVER](#).

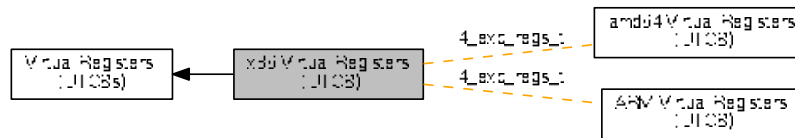
Referenced by [L4vcpu::Vcpu::wait_for_event\(\)](#).

Here is the caller graph for this function:



12.120 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Typedefs

- typedef struct [l4_exc_regs_t](#) [l4_exc_regs_t](#)
UTCB structure for exceptions.

Enumerations

- enum [L4_utcb_consts_x86](#) {
[L4_UTCB_EXCEPTION_REGS_SIZE](#) = 19, [L4_UTCB_GENERIC_DATA_SIZE](#) = 63, [L4_UTCB_GENERIC_BUFFERS_SIZE](#) = 58, [L4_UTCB_MSG_REGS_OFFSET](#) = 0,
[L4_UTCB_BUF_REGS_OFFSET](#) = 64 * sizeof([l4_umword_t](#)), [L4_UTCB_THREAD_REGS_OFFSET](#) = 123
* sizeof([l4_umword_t](#)), [L4_UTCB_INHERIT_FPU](#) = 1UL << 24, [L4_UTCB_OFFSET](#) = 512 }
UTCB constants for x86.

12.120.1 Detailed Description

12.120.2 Enumeration Type Documentation

12.120.2.1 L4_utcb_consts_x86

```
enum L4\_utcb\_consts\_x86
```

UTCB constants for x86.

Enumerator

L4_UTCB_EXCEPTION_REGS_SIZE	Number if message registers used for exception IPC.
L4_UTCB_GENERIC_DATA_SIZE	Total number of message register (MRs) available.
L4_UTCB_GENERIC_BUFFERS_SIZE	Total number of buffer registers (BRs) available.
L4_UTCB_MSG_REGS_OFFSET	Offset of MR[0] relative to the UTCB pointer.
L4_UTCB_BUF_REGS_OFFSET	Offset of BR[0] relative to the UTCB pointer.
L4_UTCB_THREAD_REGS_OFFSET	Offset of TCR[0] relative to the UTCB pointer.
L4_UTCB_INHERIT_FPU	BDR flag to accept reception of FPU state.
L4_UTCB_OFFSET	Offset of two consecutive UTCBs.

Definition at line 41 of file [utcb.h](#).

Chapter 13

Namespace Documentation

13.1 cxx Namespace Reference

Our C++ library.

Namespaces

- [Bits](#)
Internal helpers for the cxx package.

Data Structures

- class [Auto_ptr](#)
Smart pointer with automatic deletion.
- class [Avl_map](#)
AVL tree based associative container.
- class [Avl_set](#)
AVL set for simple comparable items.
- class [Avl_tree](#)
A generic AVL tree.
- class [Avl_tree_node](#)
Node of an AVL tree.
- class [Base_slab](#)
Basic slab allocator.
- class [Base_slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [Bitfield](#)
Definition for a member (part) of a bit field.
- class [Bitmap](#)
A static bit map.
- class [Bitmap_base](#)
Basic bitmap abstraction.
- class [H_list](#)
General double-linked list of unspecified [cxx::H_list_item](#) elements.

- class [H_list_item_t](#)
Basic element type for a double-linked [H_list](#).
- struct [H_list_t](#)
Double-linked list of typed [H_list_item_t](#) elements.
- class [List](#)
Doubly linked list, with internal allocation.
- class [List_alloc](#)
Standard list-based allocator.
- class [List_item](#)
Basic list item.
- struct [Lt_functor](#)
Generic comparator class that defaults to the less-than operator.
- class [New_allocator](#)
Standard allocator based on `operator new ()` .
- class [Nothrow](#)
Helper type to distinguish the `operator new` version that does not throw exceptions.
- struct [Pair](#)
Pair of two values.
- class [Pair_first_compare](#)
Comparison functor for [Pair](#).
- struct [Ref_obj_list_item](#)
Item for list linked via [cxx::Ref_ptr](#) with default reference counting.
- class [Ref_ptr](#)
A reference-counting pointer with automatic cleanup.
- class [S_list](#)
Simple single-linked list.
- class [Slab](#)
Slab allocator for object of type `Type`.
- class [Slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [static_vector](#)
Simple encapsulation for a dynamically allocated array.
- class [String](#)
Allocation free string class with explicit length field.
- class [Weak_ref](#)
Typed weak reference to an object of type `T`.
- class [Weak_ref_base](#)
Generic (base) weak reference to some object.

Typedefs

- typedef [H_list_item_t](#)< void > [H_list_item](#)
Untyped list item.
- template<typename T >
using [Ref_ptr_list_item](#) = [Bits::Smart_ptr_list_item](#)< T, [cxx::Ref_ptr](#)< T > >
Item for list linked with [cxx::Ref_ptr](#).
- template<typename T >
using [Ref_ptr_list](#) = [Bits::Smart_ptr_list](#)< [Ref_ptr_list_item](#)< T > >
Single-linked list where elements are connected via a [cxx::Ref_ptr](#).

- `template<typename T >`
`using Unique_ptr_list_item = Bits::Smart_ptr_list_item< T, cxx::unique_ptr< T > >`
Item for list linked with cxx::unique_ptr.
- `template<typename T >`
`using Unique_ptr_list = Bits::Smart_ptr_list< Unique_ptr_list_item< T > >`
Single-linked list where elements are connected with a cxx::unique_ptr.

Functions

- `template<typename T1 >`
`T1 min (T1 a, T1 b)`
Get the minimum of a and b.
- `template<typename T1 >`
`T1 max (T1 a, T1 b)`
Get the maximum of a and b.

13.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Strings.

Various kinds of C++ utilities.

13.1.2 Typedef Documentation

13.1.2.1 H_list_item

```
typedef H_list_item_t<void> cxx::H_list_item
```

Untyped list item.

Definition at line 72 of file [hlist](#).

13.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

Data Structures

- struct [Avl_map_get_key](#)
Key-getter for [Avl_map](#).
- struct [Avl_set_get_key](#)
Internal, key-getter for [Avl_set](#) nodes.
- class [Base_avl_set](#)
Internal: AVL set with internally managed nodes.
- class [Basic_list](#)
Internal: Common functions for all head-based list implementations.
- class [Bst](#)
Basic binary search tree (BST).
- class [Bst_node](#)
Basic type of a node in a binary search tree (BST).
- struct [Direction](#)
The direction to go in a binary search tree.
- class [Smart_ptr_list](#)
List of smart-pointer-managed objects.
- class [Smart_ptr_list_item](#)
List item for an arbitrary item in a [Smart_ptr_list](#).

13.2.1 Detailed Description

Internal helpers for the cxx package.

13.3 L4 Namespace Reference

[L4](#) low-level kernel interface.

Namespaces

- [lpc](#)
IPC related functionality.
- [lpc_svr](#)
Helper classes for [L4::Server](#) instantiation.
- [Typeid](#)
Definition of interface data-type helpers.
- [Types](#)
[L4](#) basic type helpers for C++.

Data Structures

- class [Alloc_list](#)
A simple list-based allocator.
- class [Arm_smccc](#)
Wrapper for function calls that follow the ARM SMC/HVC calling convention.
- class [Base_exception](#)
Base class for all exceptions, thrown by the [L4Re](#) framework.
- class [Basic_registry](#)
This registry returns the corresponding server object based on the label of an [lpc_gate](#).
- class [Bounds_error](#)
Access out of bounds.
- class [Cap](#)
C++ interface for capabilities.
- class [Cap_base](#)
Base class for all kinds of capabilities.
- class [Com_error](#)
Error conditions during IPC.
- class [Debugger](#)
C++ kernel debugger API.
- class [Element_already_exists](#)
Exception for duplicate element insertions.
- class [Element_not_found](#)
Exception for a failed lookup (element not found).
- class [Exception](#)
Exception interface.
- class [Exception_tracer](#)
Back-trace support for exceptions.
- class [Factory](#)
C++ [Factory](#) interface to create kernel objects.
- class [lcu](#)
C++ [lcu](#) interface.
- class [Invalid_capability](#)
Indicates that an invalid object was invoked.
- class [lo_pager](#)
[lo_pager](#) interface.
- class [lommu](#)
Interface for IO-MMUs used for DMA remapping.
- class [IOModifier](#)
Modifier class for the IO stream.
- class [lpc_gate](#)
The C++ IPC gate interface.
- class [lirq](#)
C++ [lirq](#) interface.
- class [lirq_eoi](#)
Interface for sending an acknowledge message to an object.
- struct [lirq_handler_object](#)
Server object base class for handling IRQ messages.
- struct [lirq_mux](#)
IRQ multiplexer for shared IRQs.
- class [Kobject](#)

- Base class for all kinds of kernel objects and remote objects, referenced by capabilities.*

 - class [Kobject_2t](#)

Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).
 - struct [Kobject_3t](#)

Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).
 - struct [Kobject_demand](#)

Get the combined server-side resource requirements for all type T...
 - class [Kobject_t](#)

Helper class to create an [L4Re](#) interface class that is derived from a single base class.
 - struct [Kobject_typeid](#)

Meta object for handling access to type information of Kobjects.
 - struct [Kobject_typeid< void >](#)

Minimalistic ID for void interface.
 - struct [Kobject_x](#)

Generic [Kobject](#) inheritance template.
 - class [Meta](#)

Meta interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.
 - class [Out_of_memory](#)

Exception signalling insufficient memory.
 - class [Pager](#)

Pager interface including the [lo_pager](#) interface.
 - class [Platform_control](#)

L4 C++ interface for controlling platform-wide properties.
 - class [Poll_timeout_kipclock](#)

A polling timeout based on the [L4Re](#) clock.
 - struct [Proto_t](#)

Data type for defining protocol numbers.
 - class [Rcv_endpoint](#)

Interface for kernel objects that allow to receive IPC from them.
 - class [Registry_iface](#)

Abstract interface for object registries.
 - class [Runtime_error](#)

Exception for an abstract runtime error.
 - class [Scheduler](#)

C++ interface of the [Scheduler](#) kernel object.
 - struct [Semaphore](#)

Kernel-provided semaphore object.
 - class [Server](#)

Basic server loop for handling client requests.
 - class [Server_object](#)

Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).
 - struct [Server_object_t](#)

Base class (template) for server implementing server objects.
 - struct [Server_object_x](#)

Helper class to implement p_dispatch based server objects.
 - class [Smart_cap](#)

Smart capability class.
 - class [String](#)

A null-terminated string container class.
 - class [Task](#)

- C++ interface of the [Task](#) kernel object.*
- class [Thread](#)
 - C++ [L4](#) kernel thread interface.*
- struct [Triggerable](#)
 - Interface that allows an object to be triggered by some source.*
- struct [Type_info](#)
 - Dynamic Type Information for [L4Re](#) Interfaces.*
- class [Unknown_error](#)
 - Exception for an unknown condition.*
- class [Vcon](#)
 - C++ [L4](#) [Vcon](#) interface.*
- class [Vm](#)
 - Virtual machine.*

Typedefs

- typedef int [Opcode](#)
 - Data type for RPC opcodes.*

Enumerations

- enum { [PROTO_ANY](#) = 0, [PROTO_EMPTY](#) = -19 }

Functions

- template<typename T >
[Type_info](#) const * [kobject_typeid](#) ()
Get the [L4::Type_info](#) for the [L4Re](#) interface given in T.
- template<typename T , typename F >
[Cap](#)< T > [cap_dynamic_cast](#) ([Cap](#)< F > const &c) throw ()
dynamic_cast for capabilities.
- template<typename T , typename F >
[Cap](#)< T > [cap_cast](#) ([Cap](#)< F > const &c) throw ()
static_cast for capabilities.
- template<typename T , typename F >
[Cap](#)< T > [cap_reinterpret_cast](#) ([Cap](#)< F > const &c) throw ()
reinterpret_cast for capabilities.
- template<typename T , typename F , typename SMART >
[Smart_cap](#)< T, SMART > [cap_cast](#) ([Smart_cap](#)< F, SMART > const &c) throw ()
static_cast for (smart) capabilities.
- template<typename T , typename F , typename SMART >
[Smart_cap](#)< T, SMART > [cap_reinterpret_cast](#) ([Smart_cap](#)< F, SMART > const &c) throw ()
reinterpret_cast for (smart) capabilities.
- void [throw_ipc_exception](#) ([L4::Cap](#)< void > const &o, [l4_msgtag_t](#) const &err, [l4_utcb_t](#) *utcb)
Throw an [L4](#) IPC error as exception.
- void [throw_ipc_exception](#) (void const *o, [l4_msgtag_t](#) const &err, [l4_utcb_t](#) *utcb)
Throw an [L4](#) IPC error as exception.

Variables

- [IOModifier](#) const [hex](#)
Modifies the stream to print numbers as hexadecimal values.
- [IOModifier](#) const [dec](#)
Modifies the stream to print numbers as decimal values.
- [BasicOStream](#) [cout](#)
Standard output stream.
- [BasicOStream](#) [cerr](#)
Standard error stream.

13.3.1 Detailed Description

[L4](#) low-level kernel interface.

13.3.2 Enumeration Type Documentation

13.3.2.1 anonymous enum

anonymous enum

Enumerator

PROTO_ANY	Default protocol used by Kobject_t and Kobject_x .
PROTO_EMPTY	Empty protocol for empty APIs.

Definition at line 55 of file [__typeinfo.h](#).

13.3.3 Function Documentation

13.3.3.1 [cap_cast\(\)](#) [1/2]

```
template<typename T , typename F , typename SMART >
Smart_cap<T, SMART> L4::cap_cast (
    Smart_cap< F, SMART > const & c ) throw ()    [inline]
```

`static_cast` for (smart) capabilities.

Template Parameters

<i>T</i>	Type to cast the capability to.
<i>F</i>	(implicit) Type of the passed capability.
<i>SMART</i>	(implicit) Class implementing the Smart_cap interface.

Parameters

<code>c</code>	Capability to be casted.
----------------	--------------------------

Returns

A smart capability with new type `T`.

Definition at line 203 of file [smart_capability](#).

13.3.3.2 `cap_cast()` [2/2]

```
template<typename T , typename F >
Cap<T> L4::cap_cast (
    Cap< F > const & c ) throw ()    [inline]
```

`static_cast` for capabilities.

Template Parameters

<code>T</code>	The target type of the capability
<code>F</code>	The source type (and is usually implicitly set)

Parameters

<code>c</code>	The source capability that shall be casted
----------------	--

Returns

A capability typed to the interface `T`.

The use of this cast operator is similar to the `static_cast<>()` for C++ pointers. It does the same type checking and adjustments like C++ does on pointers.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_cast<L4::Icu>(obj);
```

Definition at line 379 of file [capability.h](#).

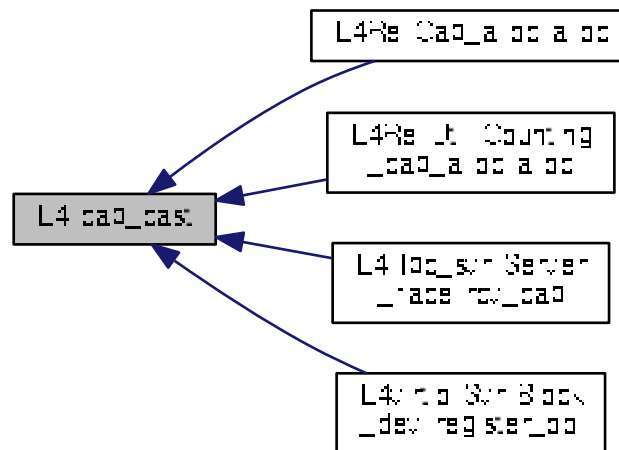
References [L4::Cap_base::cap\(\)](#).

Referenced by [L4Re::Cap_alloc::alloc\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), [L4::lpc_↔svr::Server_iface::rcv_cap\(\)](#), and [L4virtio::Svr::Block_dev< Ds_data >::register_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.3.3.3 cap_dynamic_cast()

```

template<typename T , typename F >
Cap<T> L4::cap_dynamic_cast (
    Cap< F > const & c ) throw ()    [inline]
  
```

dynamic_cast for capabilities.

Template Parameters

<i>T</i>	The target type of the capability.
<i>F</i>	The source type (is usually implicitly set).

Parameters

<code>c</code>	The source capability that shall be casted.
----------------	---

Return values

<code>Cap<T></code>	Capability of target interface <code>T</code> .
<code>L4_INVALID_CAP</code>	<code>c</code> does not support the target interface <code>T</code> or the L4::Meta interface.

The use of this cast operator is similar to the `dynamic_cast<>()` for C++ pointers. It also induces overhead, because it uses the meta interface ([L4::Meta](#)) to do runtime type checking.

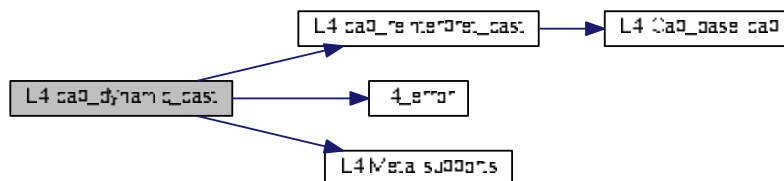
Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_dynamic_cast<L4::Icu>(obj);
```

Definition at line 125 of file [capability](#).

References [cap_reinterpret_cast\(\)](#), [l4_error\(\)](#), and [L4::Meta::supports\(\)](#).

Here is the call graph for this function:

13.3.3.4 `cap_reinterpret_cast()` [1/2]

```
template<typename T , typename F , typename SMART >
Smart_cap<T, SMART> L4::cap_reinterpret_cast (
    Smart_cap<F, SMART > const & c ) throw )    [inline]
```

`reinterpret_cast` for (smart) capabilities.

Template Parameters

<code>T</code>	Type to cast the capability to.
<code>F</code>	(implicit) Type of the passed capability.
<code>SMART</code>	(implicit) Class implementing the Smart_cap interface.

Parameters

<code>c</code>	Capability to be casted.
----------------	--------------------------

Returns

A smart capability with new type `T`.

Definition at line 222 of file [smart_capability](#).

13.3.3.5 cap_reinterpret_cast() [2/2]

```
template<typename T , typename F >
Cap<T> L4::cap_reinterpret_cast (
    Cap< F > const & c ) throw ()    [inline]
```

`reinterpret_cast` for capabilities.

Template Parameters

<code>T</code>	The target type of the capability
<code>F</code>	The source type (and is usually implicitly set)

Parameters

<code>c</code>	The source capability that shall be casted
----------------	--

Returns

A capability typed to the interface `T`.

The use of this cast operator is similar to the `reinterpret_cast<>()` for C++ pointers. It does not do any type checking or type adjustment.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_reinterpret_cast<L4::Icu>(obj);
```

Definition at line 410 of file [capability.h](#).

References [L4::Cap_base::cap\(\)](#).

Referenced by [cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.3.3.6 throw_ipc_exception() [1/2]

```

void L4::throw_ipc_exception (
    L4::Cap< void > const & o,
    l4_msgtag_t const & err,
    l4_utcb_t * utcb ) [inline]
  
```

Throw an [L4](#) IPC error as exception.

Parameters

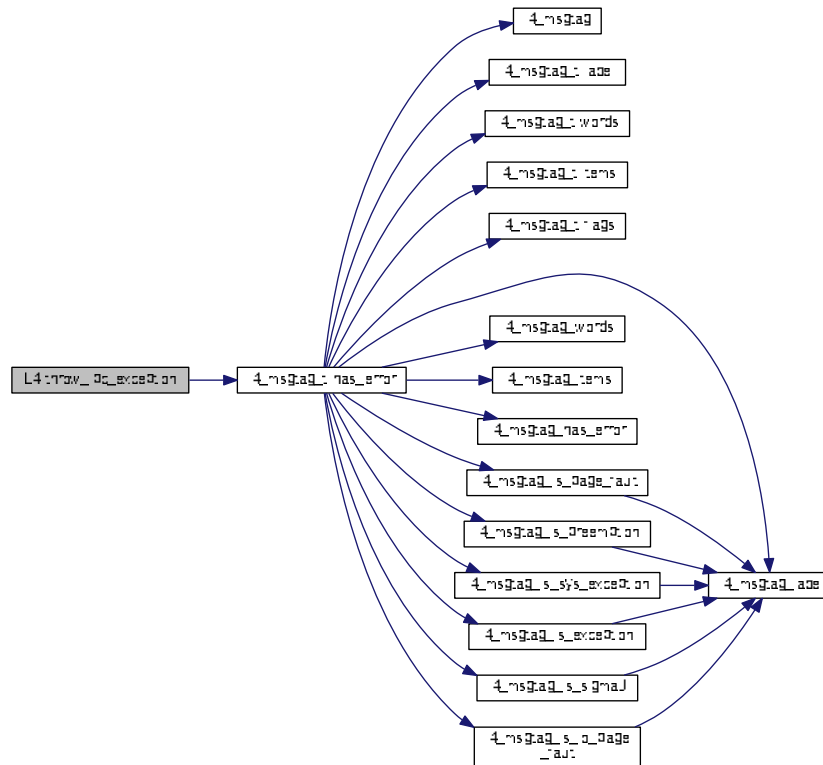
<i>o</i>	The client side object, for which the IPC was invoked.
<i>err</i>	The IPC result code (error code).
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Definition at line 41 of file [ipc_helper](#).

References [l4_msgtag_t::has_error\(\)](#).

Referenced by [throw_ipc_exception\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.3.3.7 throw_ipc_exception() [2/2]

```
void L4::throw_ipc_exception (
    void const * o,
    l4_msgtag_t const & err,
    l4_utcb_t * utcb ) [inline]
```

Throw an **L4** IPC error as exception.

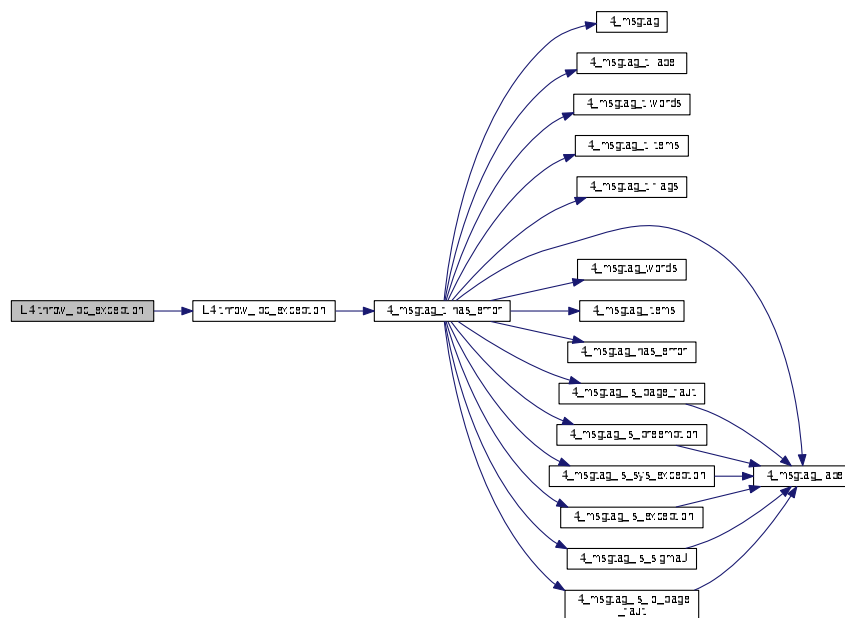
Parameters

<i>o</i>	The client side object, for which the IPC was invoked.
<i>err</i>	The IPC result code (error code).
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Definition at line 58 of file ipc_helper.

References `throw_ipc_exception()`.

Here is the call graph for this function:



13.4 L4::Ipc Namespace Reference

IPC related functionality.

Namespaces

- **Msg**

IPC Message related functionality.

Data Structures

- struct [Array](#)
Array data type for dynamically sized arrays in RPCs.
- struct [Array_in_buf](#)
Server-side copy in buffer for [Array](#).
- struct [Array_ref](#)
Array reference data type for arrays located in the message.
- struct [As_value](#)
Pass the argument as plain data value.
- class [Buf_item](#)
RPC wrapper for a receive item.
- struct [Call](#)
RPC attribute for a standard RPC call.
- struct [Call_t](#)
RPC attribute for an RPC call with required rights.
- struct [Call_zero_send_timeout](#)
RPC attribute for an RPC call, with zero send timeout.
- class [Cap](#)
Capability type for RPC interfaces (see [L4::Cap<T>](#)).
- class [Gen_fpage](#)
Generic RPC wrapper for [L4](#) flex-pages.
- struct [In_out](#)
Mark an argument as in-out argument.
- class [Iostream](#)
Input/Output stream for IPC [un]marshalling.
- class [Istream](#)
Input stream for IPC unmarshalling.
- class [Msg_ptr](#)
*Pointer to an element of type *T* in an [lpc::Istream](#).*
- struct [Opt](#)
Attribute for defining an optional RPC argument.
- class [Ostream](#)
Output stream for IPC marshalling.
- struct [Out](#)
Mark an argument as a output value in an RPC signature.
- struct [Ret_array](#)
*Dynamically sized output array of type *T*.*
- struct [Send_only](#)
RPC attribute for a send-only RPC.
- class [Small_buf](#)
A receive item for receiving a single capability.
- class [Snd_item](#)
RPC wrapper for a send item.
- class [Str_cp_in](#)
Abstraction for extracting a zero-terminated string from an [lpc::Istream](#).
- class [Varg](#)
Variably sized RPC argument.
- class [Varg_list](#)
Self-contained list of variable-sized RPC parameters.
- class [Varg_list_ref](#)
List of variable-sized RPC parameters as received by the server.

Typedefs

- typedef unsigned short [Array_len_default](#)
Default type for passing length of an array.
- typedef [Gen_fpage](#)< [Snd_item](#) > [Snd_fpage](#)
Send flex-page.
- typedef [Gen_fpage](#)< [Buf_item](#) > [Rcv_fpage](#)
Rcv flex-page.

Functions

- template<typename T >
[Cap](#)< T > [make_cap](#) ([L4::Cap](#)< T > cap, unsigned rights)
Make an L4::lpc::Cap<T> for the given capability and rights.
- template<typename T >
[Cap](#)< T > [make_cap_rw](#) ([L4::Cap](#)< T > cap)
Make an L4::lpc::Cap<T> for the given capability with [L4_CAP_FPAGE_RW](#) rights.
- template<typename T >
[Cap](#)< T > [make_cap_rws](#) ([L4::Cap](#)< T > cap)
Make an L4::lpc::Cap<T> for the given capability with [L4_CAP_FPAGE_RWS](#) rights.
- template<typename T >
[Cap](#)< T > [make_cap_full](#) ([L4::Cap](#)< T > cap)
Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.
- template<typename T >
[Internal::Buf_cp_out](#)< T > [buf_cp_out](#) (T const *v, unsigned long size)
Insert an array into an [lpc::Ostream](#).
- template<typename T >
[Internal::Buf_cp_in](#)< T > [buf_cp_in](#) (T *v, unsigned long &size)
Extract an array from an [lpc::Istream](#).
- template<typename T >
[Str_cp_in](#)< T > [str_cp_in](#) (T *v, unsigned long &size)
Create a [Str_cp_in](#) for the given values.
- template<typename T >
[Msg_ptr](#)< T > [msg_ptr](#) (T *&p)
Create an [Msg_ptr](#) to adjust the given pointer.
- template<typename T >
[Internal::Buf_in](#)< T > [buf_in](#) (T *&v, unsigned long &size)
Return a pointer to stream array data.
- template<typename T >
T [read](#) ([Istream](#) &s)
Read a value out of a stream.

13.4.1 Detailed Description

IPC related functionality.

13.4.2 Function Documentation

13.4.2.1 `buf_cp_in()`

```
template<typename T >
Internal::Buf_cp_in<T> L4::Ipc::buf_cp_in (
    T * v,
    unsigned long & size )
```

Extract an array from an [lpc::lstream](#).

Parameters

	<i>v</i>	Pointer to the array that shall receive the values from the lpc::lstream .
<i>in, out</i>	<i>size</i>	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

[buf_cp_in\(\)](#) can be used to extract an array from an [lpc::lstream](#). This is the counterpart [buf_cp_out\(\)](#). The data from the received message is thereby copied to the given buffer and size is set to the number of elements found in the stream. To avoid the copy operation [buf_in\(\)](#) may be used instead.

See also

[buf_in\(\)](#) and [buf_cp_out\(\)](#).

Definition at line 172 of file [ipc_stream](#).

13.4.2.2 `buf_cp_out()`

```
template<typename T >
Internal::Buf_cp_out<T> L4::Ipc::buf_cp_out (
    T const * v,
    unsigned long size )
```

Insert an array into an [lpc::Ostream](#).

Parameters

<i>v</i>	Pointer to the array that shall be inserted into an lpc::Ostream .
<i>size</i>	Number of elements in the array.

This function inserts an array (e.g. a string) into an [lpc::Ostream](#). The data is copied to the stream. On insertion into the [lpc::Ostream](#) exactly the given number of elements of type T are copied to the message buffer, this means the source buffer is no longer referenced after insertion into the stream.

See also

The counterpart is either [buf_cp_in\(\)](#) or [buf_in\(\)](#).

Definition at line 113 of file [ipc_stream](#).

13.4.2.3 `buf_in()`

```
template<typename T >
Internal::Buf_in<T> L4::Ipc::buf_in (
    T *& v,
    unsigned long & size )
```

Return a pointer to stream array data.

Parameters

out	<i>v</i>	Pointer to the array within the lpc::lstream .
out	<i>size</i>	The number of elements found in the stream.

This routine provides a possibility to extract an array from an [lpc::lstream](#), without extra copy overhead. In contrast to [buf_cp_in\(\)](#) the data is not copied to a buffer, but a pointer to the array is returned. The user must make sure the UTCB is not used for other purposes while the returned pointer is still in use.

The mechanism is comparable to that of [Msg_ptr](#), however it handles arrays inserted with [buf_cp_out\(\)](#).

See also

[buf_cp_in\(\)](#) and [buf_cp_out\(\)](#).

Definition at line 323 of file [ipc_stream](#).

13.4.2.4 `make_cap()`

```
template<typename T >
Cap<T> L4::Ipc::make_cap (
    L4::Cap< T > cap,
    unsigned rights )
```

Make an `L4::lpc::Cap<T>` for the given capability and rights.

Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

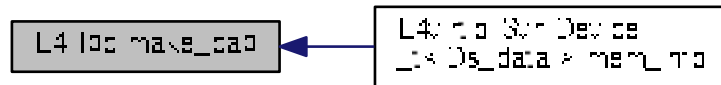
Parameters

<i>cap</i>	source capability (<code>L4::Cap<T></code>)
<i>rights</i>	rights mask that shall be applied on transfer.

Definition at line 624 of file [ipc_types](#).

Referenced by [L4virtio::Svr::Device_t< Ds_data >::mem_info\(\)](#).

Here is the caller graph for this function:



13.4.2.5 make_cap_full()

```

template<typename T >
Cap<T> L4::Ipc::make_cap_full (
    L4::Cap< T > cap )
  
```

Make an `L4::Ipc::Cap<T>` for the given capability with full fpage and object-specific rights.

Template Parameters

<i>T</i>	(implicit) type of the referenced interface
----------	---

Parameters

<i>cap</i>	source capability (<code>L4::Cap<T></code>)
------------	---

See also

[L4_cap_fpage_rights](#)
[L4_obj_fpage_ctl](#)

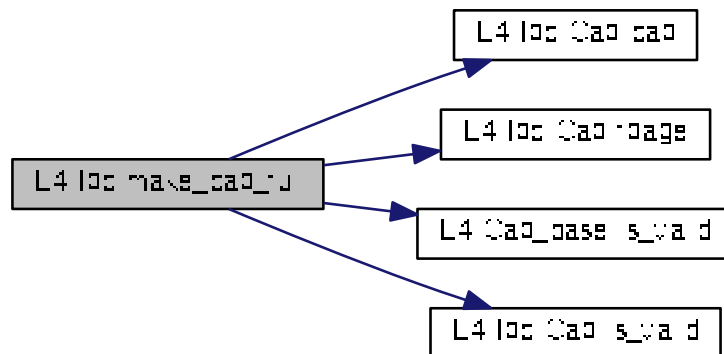
Note

Full rights (including object-specific rights) are required when mapping an IPC gate where the receiver should become the server, i.e. where the receiver wants to call [L4::ipc_gate::bind_thread\(\)](#).

Definition at line 662 of file [ipc_types](#).

References [L4::ipc::Cap< T >::cap\(\)](#), [L4::ipc::Cap< T >::fpage\(\)](#), [L4::Cap_base::is_valid\(\)](#), [L4::ipc::Cap< T >::is_valid\(\)](#), [L4_CAP_FPAGE_RWSD](#), [L4_EMSGMISSARG](#), [L4_FPAGE_C_OBJ_RIGHTS](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



13.4.2.6 make_cap_rw()

```

template<typename T >
Cap<T> L4::Ipc::make_cap_rw (
    L4::Cap< T > cap )
  
```

Make an `L4::Ipc::Cap<T>` for the given capability with `L4_CAP_FPAGE_RW` rights.

Template Parameters

<code>T</code>	(IMPLICIT) type of the referenced interface
----------------	---

Parameters

<code>cap</code>	source capability (<code>L4::Cap<T></code>)
------------------	---

Examples:

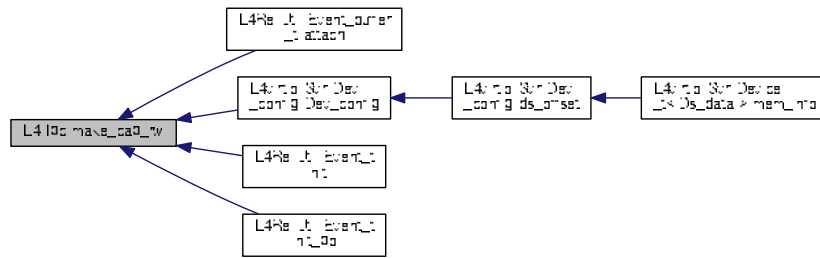
[examples/libs/l4re/c++/mem_alloc/marm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/c++/shared+_ds/ds_srv.cc](#).

Definition at line 634 of file `ipc_types`.

References [L4_CAP_FPAGE_RW](#).

Referenced by `L4Re::Util::Event_buffer_t< PAYLOAD >::attach()`, `L4virtio::Svr::Dev_config::Dev_config()`, `L4Re::Util::Event_t< PAYLOAD >::init()`, and `L4Re::Util::Event_t< PAYLOAD >::init_poll()`.

Here is the caller graph for this function:



13.4.2.7 make_cap_rws()

```
template<typename T >
Cap<T> L4::Ipc::make_cap_rws (
    L4::Cap< T > cap )
```

Make an `L4::Ipc::Cap<T>` for the given capability with `L4_CAP_FPAGE_RWS` rights.

Template Parameters

<code>T</code>	(IMPLICIT) type of the referenced interface
----------------	---

Parameters

<code>cap</code>	source capability (<code>L4::Cap<T></code>)
------------------	---

Definition at line 644 of file `ipc_types`.

References `L4_CAP_FPAGE_RWS`.

13.4.2.8 msg_ptr()

```
template<typename T >
Msg_ptr<T> L4::Ipc::msg_ptr (
    T *& p )
```

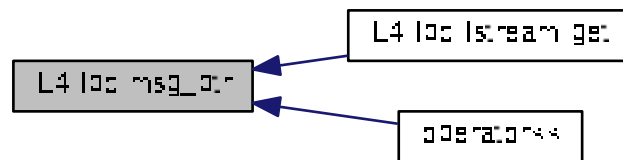
Create an `Msg_ptr` to adjust the given pointer.

This function makes it more convenient to extract pointers to data in the message buffer itself from an `Ipc::Istream`. This may be used to avoid copy out of large data structures. (See `Msg_ptr`.)

Definition at line 265 of file `ipc_stream`.

Referenced by [L4::ipc::Istream::get\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:



13.4.2.9 read()

```
template<typename T >
T L4::Ipc::read (
    Istream & s ) [inline]
```

Read a value out of a stream.

Parameters

<i>s</i>	An Istream .
----------	------------------------------

Returns

The value of type *T*.

The stream position is progressed accordingly.

Definition at line [1394](#) of file [ipc_stream](#).

13.4.2.10 str_cp_in()

```
template<typename T >
Str_cp_in<T> L4::Ipc::str_cp_in (
    T * v,
    unsigned long & size )
```

Create a [Str_cp_in](#) for the given values.

Parameters

	<i>v</i>	Pointer to the array that shall receive the values from the lpc::lstream .
<i>in, out</i>	<i>size</i>	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

This function makes it more convenient to extract arrays from an [lpc::lstream](#) (

See also

[Str_cp_in](#).)

Definition at line 226 of file [ipc_stream](#).

13.5 L4::lpc::Msg Namespace Reference

IPC Message related functionality.

Data Structures

- struct [Clnt_val_ops](#)
Defines client-side handling of 'MTYPE' as RPC argument.
- struct [Cls_buffer](#)
Marker type for receive buffer values.
- struct [Cls_data](#)
Marker type for data values.
- struct [Cls_item](#)
Marker type for item values.
- struct [Dir_in](#)
Marker type for input values.
- struct [Dir_out](#)
Marker type for output values.
- struct [Do_in_data](#)
Marker for Input data.
- struct [Do_in_items](#)
Marker for Input items.
- struct [Do_out_data](#)
Marker for Output data.
- struct [Do_out_items](#)
Marker for Output items.
- struct [Do_rcv_buffers](#)
Marker for receive buffers.
- struct [Elem< Array< A, LEN > &>](#)
Array as output argument.
- struct [Elem< Array< A, LEN > >](#)
Array as input arguments.
- struct [Elem< Array_ref< A, LEN > &>](#)
Array_ref as output argument.

- struct [ls_valid_rpc_type](#)
Type trait defining a valid RPC parameter type.
- struct [Svr_arg_pack](#)
Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.
- struct [Svr_val_ops](#)
Defines server-side handling for `MTYPE` server arguments.

Enumerations

- enum {
[Word_bytes](#) = sizeof(l4_umword_t), [Item_words](#) = 2, [Item_bytes](#) = Word_bytes * Item_words, [Mr_words](#) = L4_UTCB_GENERIC_DATA_SIZE,
[Mr_bytes](#) = Word_bytes * Mr_words, [Br_bytes](#) = Word_bytes * L4_UTCB_GENERIC_BUFFERS_SIZE }

Functions

- unsigned long [align_to](#) (unsigned long bytes, unsigned long align)
Pad bytes to the given alignment align (in bytes)
- template<typename T >
 unsigned long [align_to](#) (unsigned long bytes)
Pad bytes to the alignment of the type T.
- template<typename T >
 bool [check_size](#) (unsigned offset, unsigned limit)
Check if there is enough space for T from offset to limit.
- template<typename T , typename CTYPE >
 bool [check_size](#) (unsigned offset, unsigned limit, CTYPE cnt)
Check if there is enough space for an array of T from offset to limit.
- template<typename T >
 int [msg_add](#) (char *msg, unsigned offs, unsigned limit, T v)
Add some data to a message at offs.
- template<typename T >
 int [msg_get](#) (char *msg, unsigned offs, unsigned limit, T &v)
Get some data from a message at offs.

13.5.1 Detailed Description

IPC Message related functionality.

13.5.2 Enumeration Type Documentation

13.5.2.1 anonymous enum

anonymous enum

Enumerator

Word_bytes	number of bytes for one message word
Item_words	number of message words for one message item
Item_bytes	number of bytes for one message item
Mr_words	number of message words available in the UTCB
Mr_bytes	number of bytes available in the UTCB message registers
Br_bytes	number of bytes available in the UTCB buffer registers

Definition at line 96 of file [ipc_basics](#).

13.5.3 Function Documentation

13.5.3.1 align_to() [1/2]

```
unsigned long L4::Ipc::Msg::align_to (
    unsigned long bytes,
    unsigned long align ) [inline]
```

Pad bytes to the given alignment *align* (in bytes)

Parameters

<i>bytes</i>	The input value in bytes
<i>align</i>	The alignment value in bytes

Returns

the result after padding *bytes* to *align*.

Definition at line 41 of file [ipc_basics](#).

Referenced by [align_to\(\)](#).

Here is the caller graph for this function:



13.5.3.2 align_to() [2/2]

```
template<typename T >
unsigned long L4::IpC::Msg::align_to (
    unsigned long bytes ) [inline]
```

Pad *bytes* to the alignment of the type *T*.

Template Parameters

<i>T</i>	The data type used for the alignment
----------	--------------------------------------

Parameters

<i>bytes</i>	The value to add the padding to
--------------	---------------------------------

Returns

bytes padded to achieve the alignment of *T*.

Definition at line 51 of file [ipc_basics](#).

References [align_to\(\)](#).

Here is the call graph for this function:



13.5.3.3 check_size() [1/2]

```
template<typename T >
bool L4::IpC::Msg::check_size (
    unsigned offset,
    unsigned limit ) [inline]
```

Check if there is enough space for *T* from *offset* to *limit*.

Template Parameters

<i>T</i>	The data type that shall be fitted at <i>offset</i>
----------	---

Parameters

<i>offset</i>	The current offset in bytes (must already be padded if desired).
<i>limit</i>	The limit in bytes that must not be exceeded after adding the size of <i>T</i> .

Returns

true if the limit will not be exceeded, false else.

Definition at line 64 of file [ipc_basics](#).

13.5.3.4 `check_size()` [2/2]

```
template<typename T , typename CTYPE >
bool L4::Ipc::Msg::check_size (
    unsigned offset,
    unsigned limit,
    CTYPE cnt ) [inline]
```

Check if there is enough space for an array of *T* from offset to limit.

Template Parameters

<i>T</i>	The data type that shall be fitted at <i>offset</i>
<i>CTYPE</i>	Type of the <i>cnt</i> parameter

Parameters

<i>offset</i>	The current offset in bytes (must already be padded if desired).
<i>limit</i>	The limit in bytes that must not be exceeded after adding <i>cnt</i> times the size of <i>T</i> .
<i>cnt</i>	The number of elements of type <i>T</i> that shall be put at <i>offset</i> .

Returns

true if the limit will not be exceeded, false else.

Definition at line 82 of file [ipc_basics](#).

References [L4_UNLIKELY](#).

13.5.3.5 `msg_add()`

```
template<typename T >
int L4::Ipc::Msg::msg_add (
```

```

char * msg,
unsigned offs,
unsigned limit,
T v ) [inline]

```

Add some data to a message at offs.

Template Parameters

<i>T</i>	The type of the data to add
----------	-----------------------------

Parameters

<i>msg</i>	pointer to the start of the message
<i>offs</i>	The current offset within the message, this shall be padded to the alignment of <i>T</i> if <i>v</i> is added.
<i>limit</i>	The size limit in bytes that offset must not exceed.
<i>v</i>	The value to add to the message

Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 125 of file [ipc_basics](#).

References [L4_MSGTOOLONG](#), and [L4_UNLIKELY](#).

13.5.3.6 msg_get()

```

template<typename T >
int L4::ipc::Msg::msg_get (
    char * msg,
    unsigned offs,
    unsigned limit,
    T & v ) [inline]

```

Get some data from a message at offs.

Template Parameters

<i>T</i>	The type of the data to read
----------	------------------------------

Parameters

<i>msg</i>	Pointer to the start of the message
<i>offs</i>	The current offset within the message, this shall be padded to the alignment of <i>T</i> if a <i>v</i> can be read.
<i>limit</i>	The size limit in bytes that offset must not exceed.
<i>v</i>	A reference to receive the value from the message

Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 146 of file [ipc_basics](#).

References [L4_MSGTOOSHORT](#), and [L4_UNLIKELY](#).

13.6 L4::ipc_svr Namespace Reference

Helper classes for [L4::Server](#) instantiation.

Data Structures

- class [Br_manager_no_buffers](#)
Empty implementation of [Server_iface](#).
- struct [Compound_reply](#)
Mix in for LOOP_HOOKS to always use compound reply and wait.
- struct [Default_loop_hooks](#)
Default LOOP_HOOKS.
- struct [Default_setup_wait](#)
Mix in for LOOP_HOOKS for setup_wait no op.
- struct [Default_timeout](#)
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.
- struct [Direct_dispatch](#)
Direct dispatch helper, for forwarding dispatch calls a registry R.
- struct [Direct_dispatch< R * >](#)
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry R.
- struct [Exc_dispatch](#)
Dispatch helper wrapping try {} catch {} around the dispatch call.
- struct [Ignore_errors](#)
Mix in for LOOP_HOOKS to ignore IPC errors.
- class [Server_iface](#)
Interface for server-loop related functions.
- class [Timed_work](#)
DEPRECATED.
- class [Timeout](#)
Callback interface for [Timeout_queue](#).
- class [Timeout_queue](#)
[Timeout](#) queue to be used in l4re server loop.
- class [Timeout_queue_hooks](#)
Loop hooks mixin for integrating a timeout queue into the server loop.

Enumerations

- enum [Reply_mode](#) { [Reply_compound](#), [Reply_separate](#) }
Reply mode for server loop.

13.6.1 Detailed Description

Helper classes for [L4::Server](#) instantiation.

13.7 L4::Typeid Namespace Reference

Definition of interface data-type helpers.

Data Structures

- struct [P_dispatch](#)
Use for protocol based dispatch stage.
- struct [Raw_ipc](#)
RPCs list for passing raw incoming IPC to the server object.
- struct [Rpc_nocode](#)
List of RPCs of an interface using a single operation without an opcode.
- struct [Rpc](#)
Standard list of RPCs of an interface.
- struct [Rpc_code](#)
List of RPCs of an interface using a special opcode type.
- struct [Rpc_sys](#)
List of RPCs typically used for kernel interfaces.

13.7.1 Detailed Description

Definition of interface data-type helpers.

Note

These type helpers are intended for internal use, if you look for standard C++ type traits use the `<type_traits>` header for the standard C++ library or use `<l4/cxx/type_traits>`.

13.8 L4::Types Namespace Reference

[L4](#) basic type helpers for C++.

Data Structures

- struct [Bool](#)
Boolean meta type.
- struct [False](#)
False meta value.
- class [Flags](#)
Template for defining typical [Flags](#) bitmaps.
- struct [Same](#)
Compare two data types for equality.
- struct [True](#)
True meta value.

13.8.1 Detailed Description

[L4](#) basic type helpers for C++.

13.9 L4Re Namespace Reference

[L4Re](#) C++ Interfaces.

Namespaces

- [Util](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C++.
- [Vfs](#)
Virtual file system for interfaces POSIX libc.

Data Structures

- class [Cap_alloc](#)
Capability allocator interface.
- class [Console](#)
[Console](#) class.
- class [Dataspace](#)
Interface for memory-like objects.
- class [Debug_obj](#)
Debug interface.
- class [Dma_space](#)
DMA Address Space.
- class [Env](#)
C++ interface of the initial environment that is provided to an [L4](#) task.
- class [Event](#)
[Event](#) class.
- class [Event_buffer_t](#)
[Event](#) buffer class.
- class [Inhibitor](#)
Set of inhibitor locks, which inhibit specific actions when held.
- class [Log](#)
[Log](#) interface class.
- class [Mem_alloc](#)
Memory allocation interface.
- struct [Mmio_space](#)
Interface for memory-like address space accessible via IPC.
- class [Namespace](#)
Name-space interface.
- class [Parent](#)
[Parent](#) interface.
- class [Rm](#)
Region map.
- class [Smart_cap_auto](#)
Helper for [Auto_cap](#) and [Auto_del_cap](#).
- class [Smart_count_cap](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).

Typedefs

- `template<typename T >`
`using Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >`
Shared capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >`
Shared capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >`
Shared capability that implements automatic free and unmap+delete of the capability selector.
- `template<typename T >`
`using shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >`
Shared capability that implements automatic free and unmap+delete of the capability selector.
- `template<typename T >`
`using Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >`
Unique capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >`
Unique capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >`
Unique capability that implements automatic free and unmap+delete of the capability selector.
- `template<typename T >`
`using unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >`
Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- `long chksys (long err, char const *extra="", long ret=0)`
Generate C++ exception on error.
- `long chksys (l4_msgtag_t const &t, char const *extra="", l4_utcb_t *utcb=l4_utcb(), long ret=0)`
Generate C++ exception on error.
- `long chksys (l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra="")`
Generate C++ exception on error.
- `template<typename T >`
`T chkcap (T &&cap, char const *extra="", long err=-L4_ENOMEM)`
Check for valid capability or raise C++ exception.
- `l4_msgtag_t chkipc (l4_msgtag_t tag, char const *extra="", l4_utcb_t *utcb=l4_utcb())`
Test a message tag for IPC errors.
- `template<typename T >`
`Shared_cap< T > make_shared_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in a Shared_cap.
- `template<typename T >`
`Shared_del_cap< T > make_shared_del_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in a Shared_del_cap.
- `template<typename T >`
`Unique_cap< T > make_unique_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in an Unique_cap.
- `template<typename T >`
`Unique_del_cap< T > make_unique_del_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in an Unique_del_cap.

13.9.1 Detailed Description

[L4Re](#) C++ Interfaces.

[L4](#) Runtime Environment.

13.9.2 Typedef Documentation

13.9.2.1 Shared_cap

```
template<typename T >
using L4Re::Shared\_cap = typedef L4::Detail::Shared_cap_impl<T, Smart\_count\_cap<L4\_FP\_ALL\_SP↔ACES> >
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::↔\[Shared_cap\]\(#\)](#).

Definition at line [44](#) of file [shared_cap](#).

13.9.2.2 shared_cap

```
template<typename T >
using L4Re::shared\_cap = typedef L4::Detail::Shared_cap_impl<T, Smart\_count\_cap<L4\_FP\_ALL\_SP↔ACES> >
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared_cap](#).

Definition at line 47 of file [shared_cap](#).

13.9.2.3 Shared_del_cap

```
template<typename T >
using L4Re::Shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DE←
LETE_OBJ> >
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared_del_cap](#).

Definition at line 80 of file [shared_cap](#).

13.9.2.4 shared_del_cap

```
template<typename T >
using L4Re::shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DE←
LETE_OBJ> >
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::↵
Shared_del_cap](#).

Definition at line 83 of file [shared_cap](#).

13.9.2.5 Unique_cap

```
template<typename T >
using L4Re::Unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES> >
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::↵
Unique_cap](#).

Definition at line 42 of file [unique_cap](#).

13.9.2.6 unique_cap

```
template<typename T >
using L4Re::unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES> >
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::↵
Unique_cap](#).

Definition at line 45 of file [unique_cap](#).

13.9.2.7 Unique_del_cap

```
template<typename T >
using L4Re::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELE←
ETE_OBJ> >
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to `Unique_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique_del_cap](#).

Definition at line 75 of file [unique_cap](#).

13.9.2.8 unique_del_cap

```
template<typename T >
using L4Re::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELE←
ETE_OBJ> >
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to `Unique_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::unique_del_cap](#).

Definition at line 78 of file [unique_cap](#).

13.9.3 Function Documentation

13.9.3.1 chkcap()

```
template<typename T >
T L4Re::chkcap (
    T && cap,
    char const * extra = "",
    long err = -L4_ENOMEM ) [inline]
```

Check for valid capability or raise C++ exception.

Template Parameters

<i>T</i>	Type of object to check, must be capability-like (L4::Cap , L4Re::Util::Unique_cap etc.)
----------	---

Parameters

<i>cap</i>	Capability value to check.
<i>extra</i>	Optional text for exception.
<i>err</i>	Error value for exception or 0 if the capability value should be used.

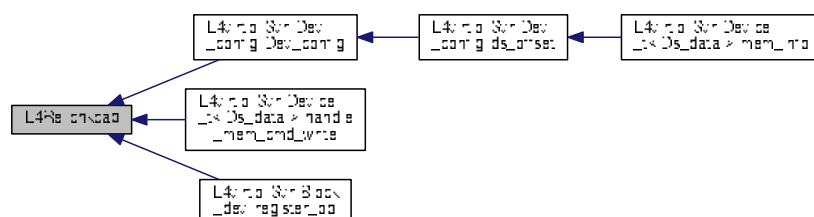
This function checks whether the capability is valid. If the capability is invalid an C++ exception is generated, using *err* if *err* is not zero, otherwise the capability value is used. A valid capability will just be returned.

Definition at line 140 of file [error_helper](#).

References [L4_UNLIKELY](#).

Referenced by [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4virtio::Svr::Device_t<Ds_data>::handle_mem_cmd_write\(\)](#), and [L4virtio::Svr::Block_dev<Ds_data>::register_obj\(\)](#).

Here is the caller graph for this function:



13.9.3.2 `chkipc()`

```
l4_msgtag_t L4Re::chkipc (
    l4_msgtag_t tag,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb() ) [inline]
```

Test a message tag for IPC errors.

Parameters

<i>tag</i>	Message tag returned by the IPC.
<i>extra</i>	Exception message in case of error.
<i>utcb</i>	The UTCB used in the IPC operation.

Returns

On IPC error an exception is thrown, otherwise `tag` is returned.

Exceptions

<i>L4::Runtime_exception</i>	with the translated IPC error code
------------------------------	------------------------------------

This function does not check the message tag's label value.

Note

This must be called on a message tag before the UTCB is changed.

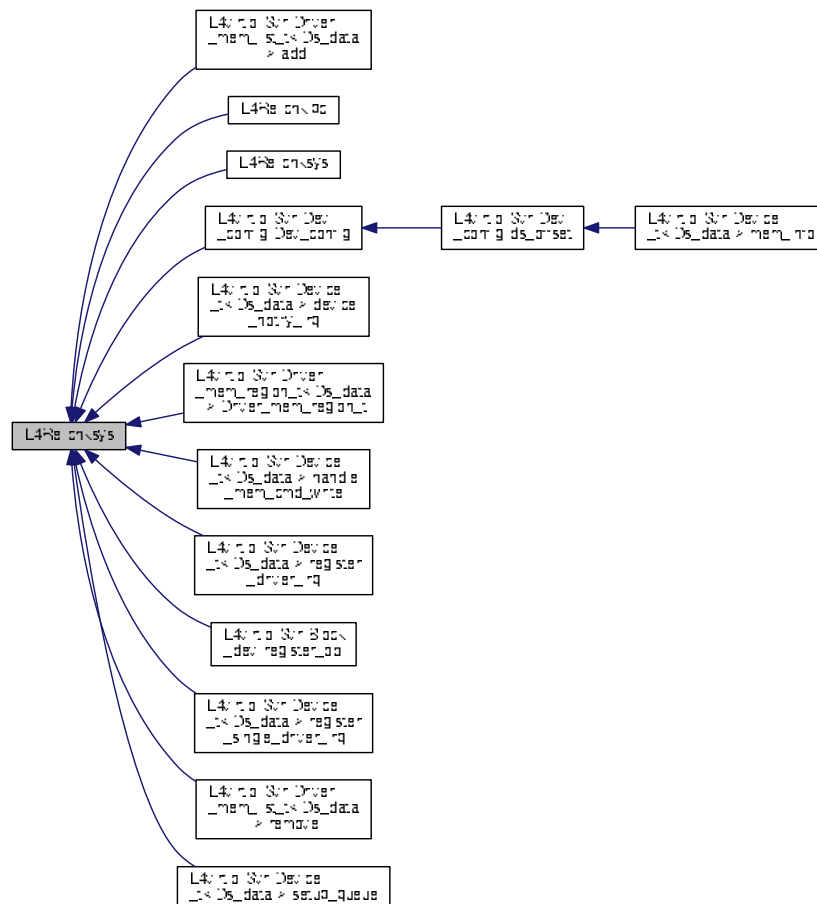
Definition at line 171 of file [error_helper](#).

References [chksys\(\)](#), [l4_msgtag_t::has_error\(\)](#), and [L4_UNLIKELY](#).

References [L4_UNLIKELY](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t<Ds_data>::add\(\)](#), [chkipc\(\)](#), [chksys\(\)](#), [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4virtio::Svr::Device_t<Ds_data>::device_notify_irq\(\)](#), [L4virtio::Svr::Driver_mem_region_t<Ds_data>::Driver_mem_region_t\(\)](#), [L4virtio::Svr::Device_t<Ds_data>::handle_mem_cmd_write\(\)](#), [L4virtio::Svr::Device_t<Ds_data>::register_driver_irq\(\)](#), [L4virtio::Svr::Block_dev<Ds_data>::register_obj\(\)](#), [L4virtio::Svr::Device_t<Ds_data>::register_single_driver_irq\(\)](#), [L4virtio::Svr::Driver_mem_list_t<Ds_data>::remove\(\)](#), and [L4virtio::Svr::Device_t<Ds_data>::setup_queue\(\)](#).

Here is the caller graph for this function:



13.9.3.4 `chksys()` [2/3]

```

long L4Re::chksys (
    l4_msgtag_t const & t,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb(),
    long ret = 0 ) [inline]

```

Generate C++ exception on error.

Parameters

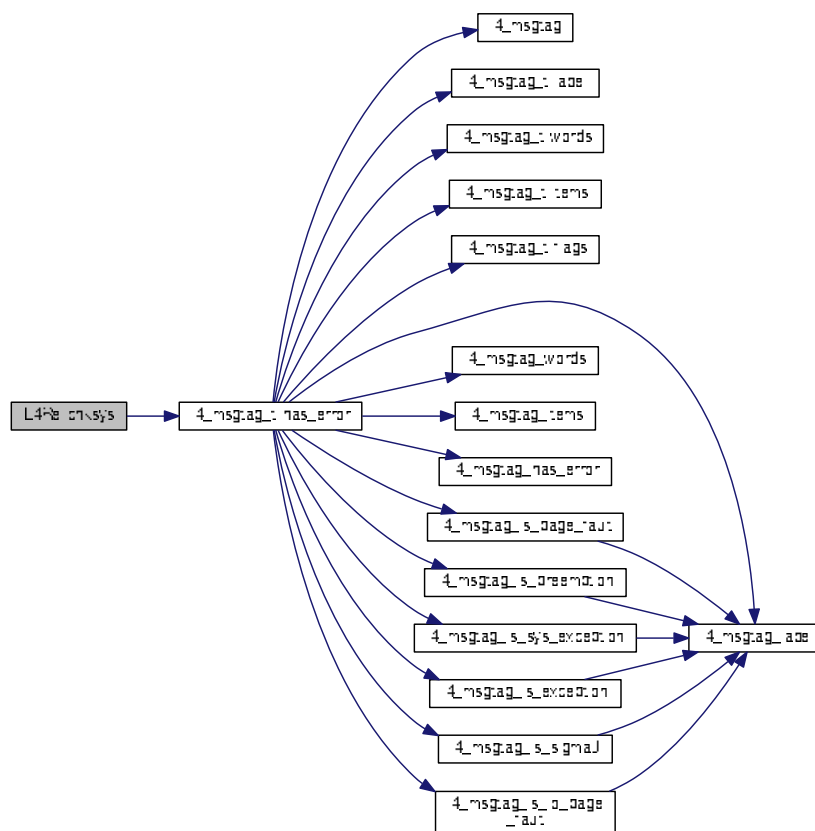
<i>t</i>	Message tag.
<i>extra</i>	Optional text for exception (default "")
<i>utcb</i>	Option UTCB
<i>ret</i>	Optional value for exception, default is error value (err)

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 84 of file [error_helper](#).

References [l4_msgtag_t::has_error\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



13.9.3.5 chksys() [3/3]

```

long L4Re::chksys (
    l4_msgtag_t const & t,
    l4_utcb_t * utcb,
    char const * extra = "" ) [inline]

```

Generate C++ exception on error.

Parameters

<i>t</i>	Message tag.
<i>utcb</i>	UTCB.
<i>extra</i>	Optional text for exception (default "")

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 107 of file [error_helper](#).

References [chksys\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:

13.9.3.6 `make_shared_cap()`

```
template<typename T >
Shared_cap<T> L4Re::make_shared_cap (
    L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in a `Shared_cap`.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_shared_cap<T>\(\)](#).

Definition at line 60 of file [shared_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



13.9.3.7 make_shared_del_cap()

```
template<typename T >
Shared_del_cap<T> L4Re::make_shared_del_cap (
    L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in a Shared_del_cap.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_shared_del_cap<T>\(\)](#).

Definition at line 96 of file [shared_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



13.9.3.8 make_unique_cap()

```
template<typename T >
Unique_cap<T> L4Re::make_unique_cap (
    L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in an Unique_cap.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_unique_cap<T>\(\)](#).

Definition at line 58 of file [unique_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



13.9.3.9 make_unique_del_cap()

```
template<typename T >
Unique_del_cap<T> L4Re::make_unique_del_cap (
    L4Re::Cap_alloc * ca )
```

Allocate a capability slot and wrap it in an Unique_del_cap.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_unique_del_cap<T>\(\)](#).

Definition at line 91 of file [unique_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



13.10 L4Re::Util Namespace Reference

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Data Structures

- struct [Auto_cap](#)
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [Auto_del_cap](#)
Automatic capability that implements automatic free and unmap+delete of the capability selector.
- class [Br_manager](#)
Buffer-register (BR) manager for [L4::Server](#).
- struct [Br_manager_hooks](#)
Predefined server-loop hooks for a server loop using the [Br_manager](#).
- struct [Br_manager_timeout_hooks](#)
Predefined server-loop hooks for a server with using the [Br_manager](#) and a timeout queue.
- class [Cap_alloc_base](#)
Capability allocator.
- struct [Counter](#)
Counter for [Counting_cap_alloc](#) with variable data width.
- class [Counting_cap_alloc](#)
Internal reference-counting cap allocator.
- class [Dataspace_svr](#)
[Dataspace](#) server class.
- class [Event_buffer_consumer_t](#)

- An event buffer consumer.*
- class [Event_buffer_t](#)
Event_buffer utility class.
- class [Event_svr](#)
Convenience wrapper for implementing an event server.
- class [Event_t](#)
Convenience wrapper for getting access to an event object.
- class [Item_alloc_base](#)
Item allocator.
- class [Object_registry](#)
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.
- struct [Ref_cap](#)
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [Ref_del_cap](#)
Automatic capability that implements automatic free and unmap+delete of the capability selector.
- class [Registry_server](#)
A server loop object which has a [Object_registry](#) included.
- class [Smart_cap_auto](#)
Helper for [Auto_cap](#) and [Auto_del_cap](#).
- class [Smart_count_cap](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).
- class [Vcon_svr](#)
[Console](#) server template class.

Typedefs

- `template<typename T >`
`using Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >`
Shared capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES > >`
Shared capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >`
Shared capability that implements automatic free and unmap+delete of the capability selector.
- `template<typename T >`
`using shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ > >`
Shared capability that implements automatic free and unmap+delete of the capability selector.
- `template<typename T >`
`using Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >`
Unique capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES > >`
Unique capability that implements automatic free and unmap of the capability selector.
- `template<typename T >`
`using Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >`
Unique capability that implements automatic free and unmap+delete of the capability selector.
- `template<typename T >`
`using unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE_OBJ > >`
Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- `template<typename T >`
`Auto_cap< T >::Cap make_auto_cap ()`
Allocate a capability slot and wrap it in an [Auto_cap](#).
- `template<typename T >`
`Auto_del_cap< T >::Cap make_auto_del_cap ()`
Allocate a capability slot and wrap it in an [Auto_del_cap](#).
- `template<typename T >`
`Ref_cap< T >::Cap make_ref_cap ()`
Allocate a capability slot and wrap it in a [Ref_cap](#).
- `template<typename T >`
`Ref_del_cap< T >::Cap make_ref_del_cap ()`
Allocate a capability slot and wrap it in a [Ref_del_cap](#).
- `int kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) throw ()`
Allocate state area.
- `template<typename T >`
`Shared_cap< T > make_shared_cap ()`
Allocate a capability slot and wrap it in a [Shared_cap](#).
- `template<typename T >`
`Shared_del_cap< T > make_shared_del_cap ()`
Allocate a capability slot and wrap it in a [Shared_del_cap](#).
- `template<typename T >`
`Unique_cap< T > make_unique_cap ()`
Allocate a capability slot and wrap it in an [Unique_cap](#).
- `template<typename T >`
`Unique_del_cap< T > make_unique_del_cap ()`
Allocate a capability slot and wrap it in an [Unique_del_cap](#).

Variables

- `_Cap_alloc & cap_alloc`
Capability allocator.

13.10.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

13.10.2 Typedef Documentation

13.10.2.1 Shared_cap

```
template<typename T >
using L4Re::Util::Shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_↔
ALL_SPACES> >
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_cap<L4Re::Dataspace>
        ds_cap = make_shared_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 59 of file [shared_cap](#).

13.10.2.2 shared_cap

```
template<typename T >
using L4Re::Util::shared_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_↔
ALL_SPACES> >
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_cap<L4Re::Dataspace>
        ds_cap = make_shared_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
```

```

    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).

```

Definition at line 62 of file [shared_cap](#).

13.10.2.3 Shared_del_cap

```

template<typename T >
using L4Re::Util::Shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4↔
_FP_DELETE_OBJ> >

```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to Shared_cap is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```

L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_del_cap<L4Re::Dataspace>
        ds_cap = make_shared_del_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).

```

Definition at line 109 of file [shared_cap](#).

13.10.2.4 shared_del_cap

```

template<typename T >
using L4Re::Util::shared_del_cap = typedef L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4↔
_FP_DELETE_OBJ> >

```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to `Shared_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_del_cap<L4Re::Dataspace>
        ds_cap = make_shared_del_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 112 of file [shared_cap](#).

13.10.2.5 Unique_cap

```
template<typename T >
using L4Re::Util::Unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_A↔
LL_SPACES> >
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

Usage:

```
{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));
```

```

...

// At the end of the scope ds_cap is unmapped and the capability
// selector is freed.
}

```

Definition at line 54 of file [unique_cap](#).

13.10.2.6 unique_cap

```

template<typename T >
using L4Re::Util::unique_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_A↔
LL_SPACES> >

```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The ownership of the capability is managed in the same way as `unique_ptr`.

Usage:

```

{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed.
}

```

Definition at line 57 of file [unique_cap](#).

13.10.2.7 Unique_del_cap

```

template<typename T >
using L4Re::Util::Unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_↔
FP_DELETE_OBJ> >

```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

The main difference to `Unique_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
        ds_cap = make_unique_del_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed. Because the deletion flag is set the data space
    // shall also be deleted (even if there are other references to this
    // data space).
}
```

Definition at line 97 of file `unique_cap`.

13.10.2.8 unique_del_cap

```
template<typename T >
using L4Re::Util::unique_del_cap = typedef L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_↵
FP_DELETE_OBJ> >
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

<code>T</code>	Type of the object the capability refers to.
----------------	--

The main difference to `Unique_cap` is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
        ds_cap = make_unique_del_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed. Because the deletion flag is set the data space
    // shall also be deleted (even if there are other references to this
    // data space).
}
```

Definition at line 100 of file `unique_cap`.

13.10.3 Function Documentation

13.10.3.1 `make_shared_cap()`

```
template<typename T >
Shared_cap<T> L4Re::Util::make_shared_cap ( )
```

Allocate a capability slot and wrap it in a `Shared_cap`.

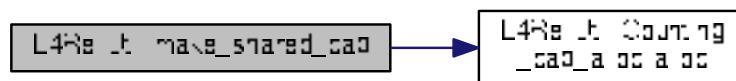
Template Parameters

<code>T</code>	Type of the object the capability refers to.
----------------	--

Definition at line 71 of file `shared_cap`.

References `L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc()`, and `cap_alloc`.

Here is the call graph for this function:



13.10.3.2 `make_shared_del_cap()`

```
template<typename T >
Shared_del_cap<T> L4Re::Util::make_shared_del_cap ( )
```

Allocate a capability slot and wrap it in a `Shared_del_cap`.

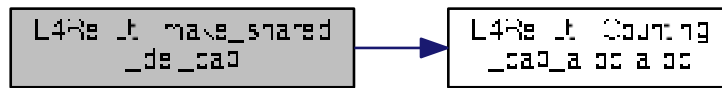
Template Parameters

<code>T</code>	Type of the object the capability refers to.
----------------	--

Definition at line 121 of file `shared_cap`.

References `L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc()`, and `cap_alloc`.

Here is the call graph for this function:



13.10.3.3 make_unique_cap()

```
template<typename T >
Unique_cap<T> L4Re::Util::make_unique_cap ( )
```

Allocate a capability slot and wrap it in an Unique_cap.

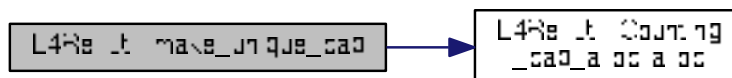
Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 66 of file [unique_cap](#).

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:



13.10.3.4 make_unique_del_cap()

```
template<typename T >
Unique_del_cap<T> L4Re::Util::make_unique_del_cap ( )
```

Allocate a capability slot and wrap it in an Unique_del_cap.

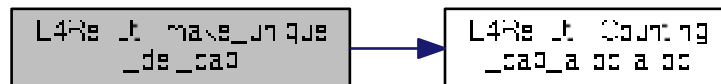
Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	--

Definition at line 109 of file [unique_cap](#).

References [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), and [cap_alloc](#).

Here is the call graph for this function:



13.11 L4Re::Vfs Namespace Reference

Virtual file system for interfaces POSIX libc.

Data Structures

- class [Be_file](#)
Boiler plate class for implementing an open file for L4Re::Vfs.
- class [Be_file_system](#)
Boilerplate class for implementing a L4Re::Vfs::File_system.
- class [Directory](#)
Interface for a POSIX file that is a directory.
- class [File](#)
The basic interface for an open POSIX file.
- class [File_system](#)
Basic interface for an L4Re::Vfs file system.
- class [Fs](#)
POSIX File-system related functionality.
- class [Generic_file](#)
The common interface for an open POSIX file.
- class [Mman](#)
Interface for the POSIX memory management.
- class [Ops](#)
Interface for the POSIX backends for an application.
- class [Regular_file](#)
Interface for a POSIX file that provides regular file semantics.
- class [Special_file](#)
Interface for a POSIX file that provides special file semantics.

Functions

- [L4Re::Vfs::Ops](#) *vfs_ops [asm](#) ("l4re_env_posix_vfs_ops")
Reference to the applications [L4Re::Vfs::Ops](#) singleton.

13.11.1 Detailed Description

Virtual file system for interfaces POSIX libc.

13.12 L4virtio Namespace Reference

L4-VIRTIO Transport C++ API.

Data Structures

- class [Device](#)
IPC interface for virtio over [L4](#) IPC.
- class [Ptr](#)
Pointer used in virtio descriptors.
- class [Virtqueue](#)
Low-level [Virtqueue](#).

13.12.1 Detailed Description

L4-VIRTIO Transport C++ API.

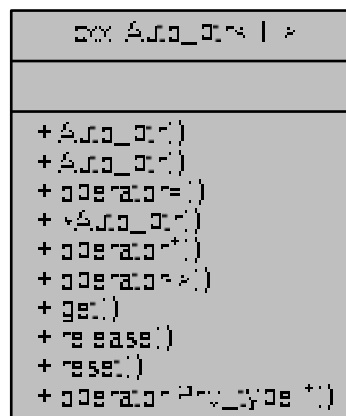
Chapter 14

Data Structure Documentation

14.1 `cxx::Auto_ptr< T >` Class Template Reference

Smart pointer with automatic deletion.

Collaboration diagram for `cxx::Auto_ptr< T >`:



Public Types

- typedef T [Ref_type](#)

The referenced type.

Public Member Functions

- [Auto_ptr](#) (T *p=0) throw ()
Construction by assignment of a normal pointer.
- [Auto_ptr](#) ([Auto_ptr](#) const &o) throw ()
Copy construction, releases the original pointer.
- [Auto_ptr](#) & [operator=](#) ([Auto_ptr](#) const &o) throw ()
Assignment from another smart pointer.
- [~Auto_ptr](#) () throw ()
Destruction, shall delete the object.
- T & [operator*](#) () const throw ()
Dereference the pointer.
- T * [operator->](#) () const throw ()
Member access for the object.
- T * [get](#) () const throw ()
Get the normal pointer.
- T * [release](#) () throw ()
Release the object and get the normal pointer back.
- void [reset](#) (T *p=0) throw ()
Delete the object and reset the smart pointer to NULL.
- [operator Priv_type *](#) () const throw ()
Operator for `if (!ptr) ...`

14.1.1 Detailed Description

```
template<typename T>
class cxx::Auto_ptr< T >
```

Smart pointer with automatic deletion.

Template Parameters

<i>T</i>	The type of the referenced object.
----------	------------------------------------

This smart pointer calls the delete operator when the destructor is called. This has the effect that the object the pointer points to will be deleted when the pointer goes out of scope, or a new value gets assigned. The smart pointer provides a [release\(\)](#) method to get a normal pointer to the object and set the smart pointer to NULL.

Definition at line 36 of file [auto_ptr](#).

14.1.2 Member Typedef Documentation

14.1.2.1 Ref_type

```
template<typename T >
typedef T cxx::Auto_ptr< T >::Ref_type
```

The referenced type.

Definition at line 41 of file [auto_ptr](#).

14.1.3 Constructor & Destructor Documentation

14.1.3.1 `Auto_ptr()` [1/2]

```
template<typename T >  
cxx::Auto_ptr< T >::Auto_ptr (  
    T * p = 0 ) throw )    [inline], [explicit]
```

Construction by assignment of a normal pointer.

Parameters

<i>p</i>	The pointer to the object
----------	---------------------------

Definition at line 51 of file `auto_ptr`.

14.1.3.2 `Auto_ptr()` [2/2]

```
template<typename T >  
cxx::Auto_ptr< T >::Auto_ptr (  
    Auto_ptr< T > const & o ) throw )    [inline]
```

Copy construction, releases the original pointer.

Parameters

<i>o</i>	The smart pointer, which shall be copied and released.
----------	--

Definition at line 57 of file `auto_ptr`.

14.1.3.3 `~Auto_ptr()`

```
template<typename T >  
cxx::Auto_ptr< T >::~~Auto_ptr ( ) throw )    [inline]
```

Destruction, shall delete the object.

Definition at line 76 of file `auto_ptr`.

14.1.4 Member Function Documentation

14.1.4.1 `get()`

```
template<typename T >
T* cxx::Auto_ptr< T >::get ( ) const throw ( ) [inline]
```

Get the normal pointer.

Attention

This function will not release the object, the object will be deleted by the smart pointer.

Definition at line 90 of file `auto_ptr`.

14.1.4.2 `operator Priv_type *()`

```
template<typename T >
cxx::Auto_ptr< T >::operator Priv_type * ( ) const throw ( ) [inline]
```

Operator for `if (!ptr)`

Definition at line 110 of file `auto_ptr`.

14.1.4.3 `operator*()`

```
template<typename T >
T& cxx::Auto_ptr< T >::operator* ( ) const throw ( ) [inline]
```

Dereference the pointer.

Definition at line 80 of file `auto_ptr`.

14.1.4.4 `operator->()`

```
template<typename T >
T* cxx::Auto_ptr< T >::operator-> ( ) const throw ( ) [inline]
```

Member access for the object.

Definition at line 83 of file `auto_ptr`.

14.1.4.5 `operator=()`

```
template<typename T >
Auto_ptr& cxx::Auto_ptr< T >::operator= (
    Auto_ptr< T > const & o ) throw ( ) [inline]
```

Assignment from another smart pointer.

Parameters

<code>o</code>	The source for the assignment (will be released).
----------------	---

Definition at line 65 of file [auto_ptr](#).

References [cxx::Auto_ptr< T >::release\(\)](#).

Here is the call graph for this function:

14.1.4.6 `release()`

```
template<typename T >
T* cxx::Auto_ptr< T >::release ( ) throw ( )    [inline]
```

Release the object and get the normal pointer back.

After calling this function the smart pointer will point to NULL and the object will not be deleted by the pointer anymore.

Definition at line 98 of file [auto_ptr](#).

Referenced by [cxx::Auto_ptr< T >::operator=\(\)](#).

Here is the caller graph for this function:



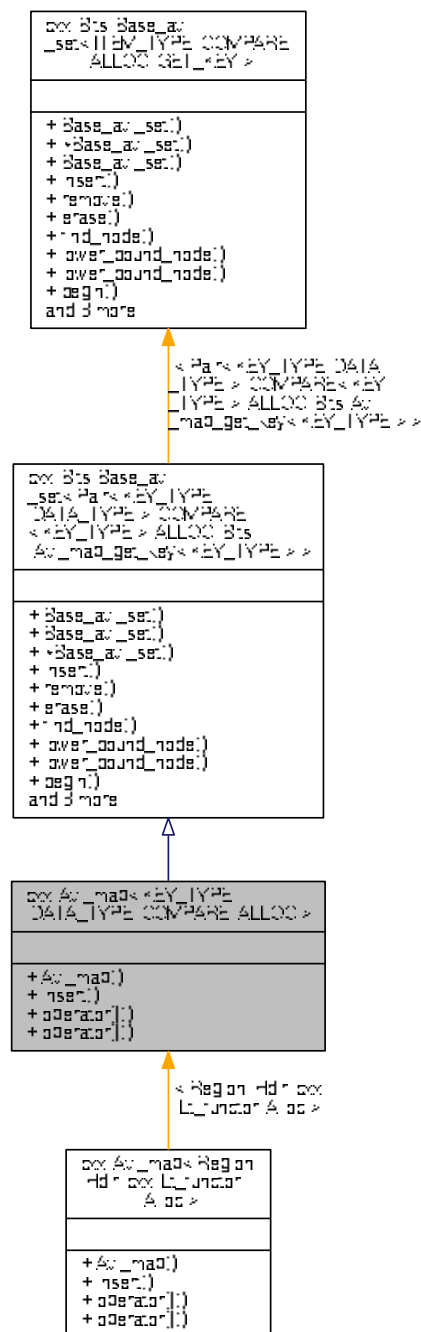
The documentation for this class was generated from the following file:

- `I4/cxx/auto_ptr`

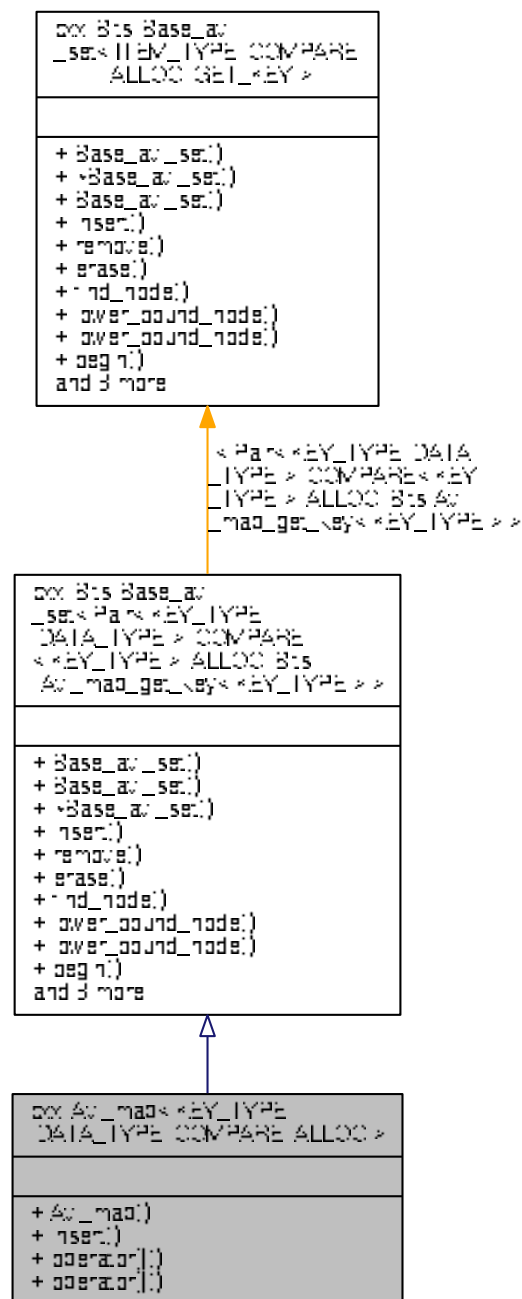
14.2 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference

AVL tree based associative container.

Inheritance diagram for cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >:



Collaboration diagram for cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >:



Public Types

- `typedef COMPARE< KEY_TYPE > Key_compare`
Type of the comparison functor.
- `typedef KEY_TYPE Key_type`
Type of the key values.
- `typedef DATA_TYPE Data_type`

Type of the data values.

- typedef [Base_type::Node](#) Node

Return type for find.

- typedef [Base_type::Node_allocator](#) Node_allocator

Type of the allocator.

Public Member Functions

- [Avl_map](#) ([Node_allocator](#) const &alloc=[Node_allocator](#)())
Create an empty AVL-tree based map.
- [cxx::Pair](#)< Iterator, int > [insert](#) ([Key_type](#) const &key, [Data_type](#) const &data)
Insert a <key, data> pair into the map.
- [Data_type](#) const & [operator\[\]](#) ([Key_type](#) const &key) const
Get the data for the given key.
- [Data_type](#) & [operator\[\]](#) ([Key_type](#) const &key)
Get or insert data for the given key.

14.2.1 Detailed Description

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_functor, template<
typename B > class ALLOC = New_allocator>
class cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >
```

AVL tree based associative container.

Template Parameters

<i>KEY_TYPE</i>	Type of the key values.
<i>DATA_TYPE</i>	Type of the data values.
<i>COMPARE</i>	Type comparison functor for the key values.
<i>ALLOC</i>	Type of the allocator used for the nodes.

Definition at line 56 of file [avl_map](#).

14.2.2 Constructor & Destructor Documentation

14.2.2.1 Avl_map()

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::Avl_map (
    Node_allocator const & alloc = Node_allocator() ) [inline]
```

Create an empty AVL-tree based map.

Parameters

<code>alloc</code>	The node allocator.
--------------------	---------------------

Definition at line 91 of file `avl_map`.

14.2.3 Member Function Documentation

14.2.3.1 `insert()`

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
cxx::Pair<Iterator, int> cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert (
    Key_type const & key,
    Data_type const & data ) [inline]
```

Insert a `<key, data>` pair into the map.

Parameters

<code>key</code>	The key value.
<code>data</code>	The data value to insert.

Returns

A pair of iterator (`first`) and return value (`second`). `second` will be 0 if the element was inserted into the set and `-E_exist` if the key was already in the set and the set was therefore not updated. In both cases, `first` contains an iterator that points to the element. `second` may also be `-E_nomem` when memory for the new node could not be allocated. `first` is then invalid.

Definition at line 110 of file `avl_map`.

14.2.3.2 `operator[]()` [1/2]

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
Data_type const& cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key ) const [inline]
```

Get the data for the given key.

Parameters

<code>key</code>	The key value to use for lookup.
------------------	----------------------------------

Precondition

A <key, data> pair for the given key value must exist.

Definition at line 118 of file [avl_map](#).

14.2.3.3 operator[]() [2/2]

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↵
functor, template< typename B > class ALLOC = New_allocator>
Data_type& cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key ) [inline]
```

Get or insert data for the given key.

Parameters

<i>key</i>	The key value to use for lookup.
------------	----------------------------------

Returns

If the item already exists, a reference to the data item. Otherwise a new data item is default-constructed and inserted under the given key before a reference is returned.

Definition at line 130 of file [avl_map](#).

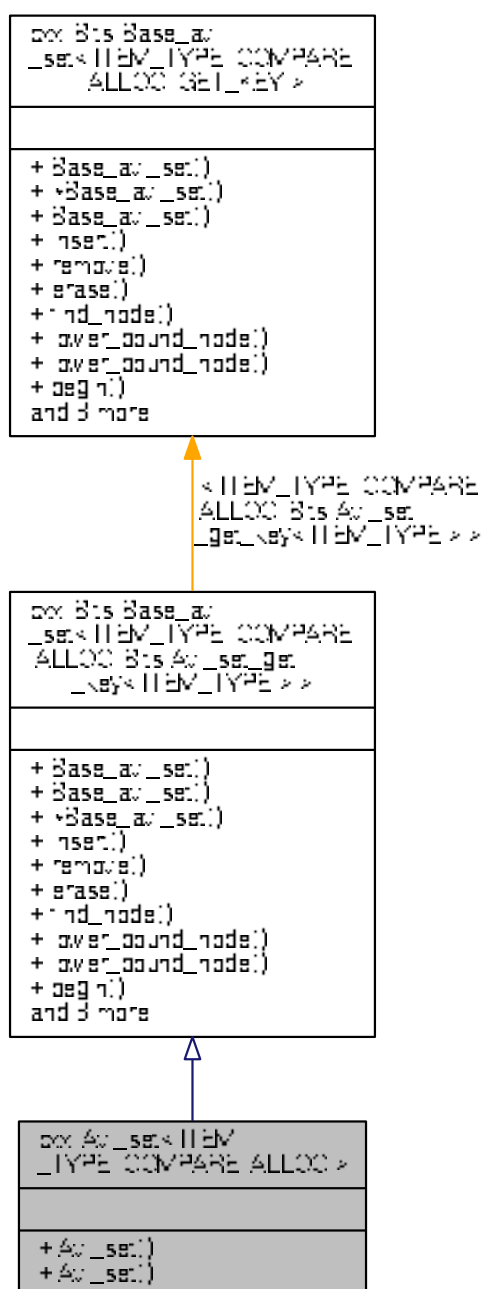
The documentation for this class was generated from the following file:

- I4/cxx/[avl_map](#)

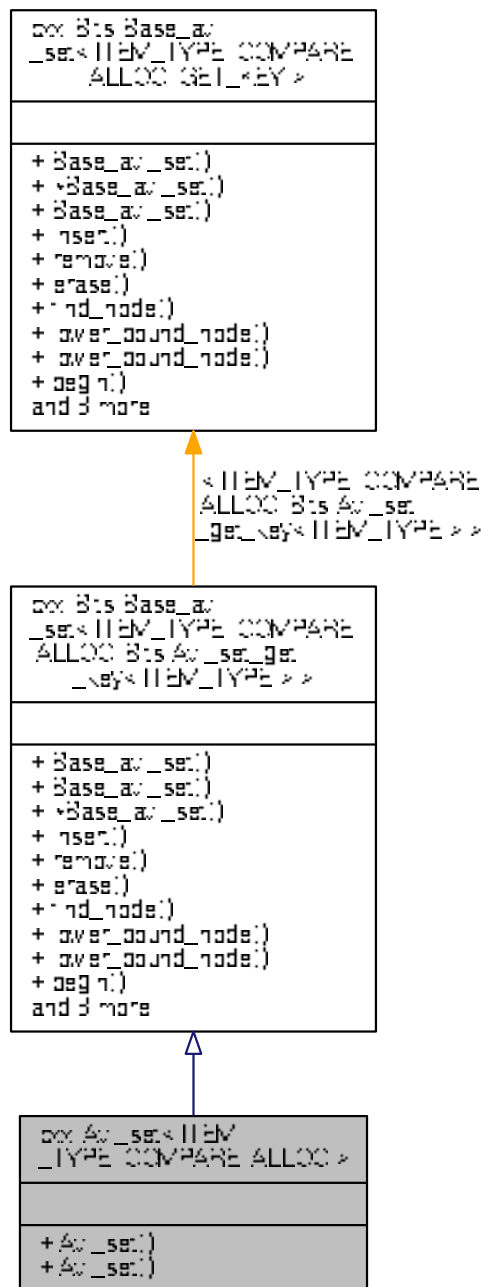
14.3 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference

AVL set for simple compareable items.

Inheritance diagram for cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >:



Collaboration diagram for `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >`:



Additional Inherited Members

14.3.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>, template< typename A > class ALLOC = New_↵  
_allocator>  
class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >
```

AVL set for simple compareable items.

The AVL set can store any kind of items where a partial order is defined. The default relation is defined by the '<' operator.

Template Parameters

<i>ITEM_TYPE</i>	The type of the items to be stored in the set.
<i>COMPARE</i>	The relation to define the partial order, default is to use operator '<'.
<i>ALLOC</i>	The allocator to use for the nodes of the AVL set.

Definition at line 429 of file [avl_set](#).

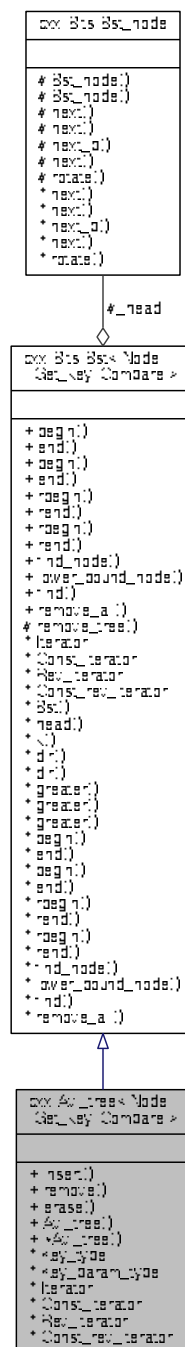
The documentation for this class was generated from the following file:

- [I4/cxx/avl_set](#)

14.4 `cxx::Avl_tree< Node, Get_key, Compare >` Class Template Reference

A generic AVL tree.

Collaboration diagram for cxx::Avl_tree< Node, Get_key, Compare >:



Public Types

- typedef [Bst::Iterator](#) `Iterator`

- Forward iterator for the tree.*
 - typedef [Bst::Const_iterator](#) [Const_iterator](#)
 - Constant forward iterator for the tree.*
- typedef [Bst::Rev_iterator](#) [Rev_iterator](#)
 - Backward iterator for the tree.*
- typedef [Bst::Const_rev_iterator](#) [Const_rev_iterator](#)
 - Constant backward iterator for the tree.*

Public Member Functions

- [Pair](#)< Node *, bool > [insert](#) (Node *new_node)
Insert a new node into this AVL tree.
- Node * [remove](#) (Key_param_type key)
Remove the node with key from the tree.
- Node * [erase](#) (Key_param_type key)
An alias for [remove\(\)](#).
- [Avl_tree](#) ()
Create an empty AVL tree.
- [~Avl_tree](#) () noexcept
Destroy the tree.

Additional Inherited Members

14.4.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare = Lt_functor<typename Get_key::Key_type>>
class cxx::Avl_tree< Node, Get_key, Compare >
```

A generic AVL tree.

Template Parameters

<i>Node</i>	The data type of the nodes (must inherit from Avl_tree_node).
<i>Get_key</i>	The meta function to get the key value from a node. The implementation uses <code>Get_key::key_of(ptr_to_node)</code> . The type of the key values must be defined in <code>Get_key::Key_type</code> .
<i>Compare</i>	Binary relation to establish a total order for the nodes of the tree. <code>Compare() (l, r)</code> must return true if the key <i>l</i> is smaller than the key <i>r</i> .

This implementation does not provide any memory management. It is the responsibility of the caller to allocate nodes before inserting them and to free them when they are removed or when the tree is destroyed. Conversely, the caller must also ensure that nodes are removed from the tree before they are destroyed.

Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 109 of file [avl_tree](#).

14.4.2 Member Function Documentation

14.4.2.1 insert()

```
template<typename Node, typename Get_key , class Compare >
Pair< Node *, bool > cxx::Avl_tree< Node, Get_key, Compare >::insert (
    Node * new_node )
```

Insert a new node into this AVL tree.

Parameters

<i>new_node</i>	A pointer to the new node.
-----------------	----------------------------

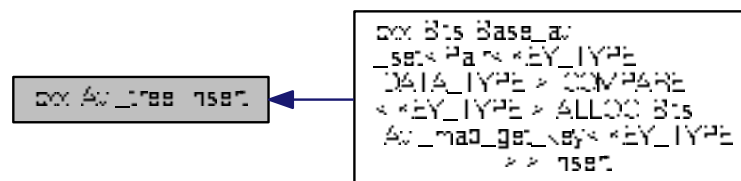
Returns

A pair, with *second* set to `true` and *first* pointing to *new_node*, on success. If there is already a node with the same key then *first* points to this node and *second* is 'false'.

Definition at line 229 of file [avl_tree](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::insert\(\)](#).

Here is the caller graph for this function:



14.4.2.2 remove()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Avl_tree< Node, Get_key, Compare >::remove (
    Key_param_type key ) [inline]
```

Remove the node with *key* from the tree.

Parameters

<i>key</i>	The key to the node to remove.
------------	--------------------------------

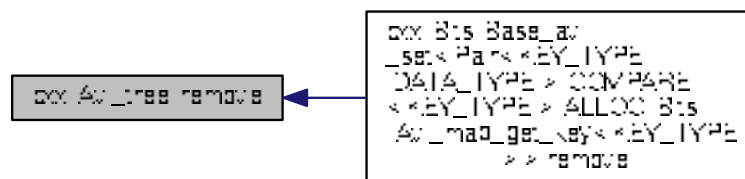
Returns

The pointer to the removed node on success, or 0 if no node with the *key* exists.

Definition at line 291 of file [avl_tree](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::remove\(\)](#).

Here is the caller graph for this function:



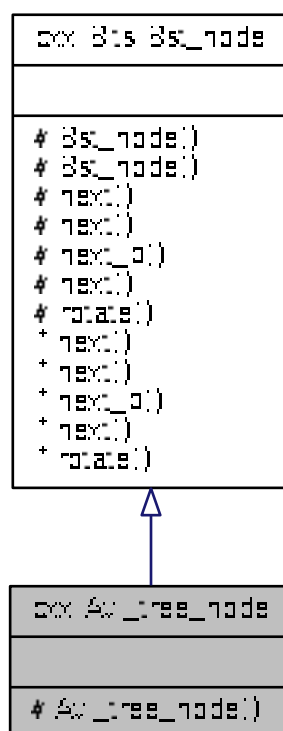
The documentation for this class was generated from the following file:

- [l4/cxx/avl_tree](#)

14.5 cxx::Avl_tree_node Class Reference

Node of an AVL tree.

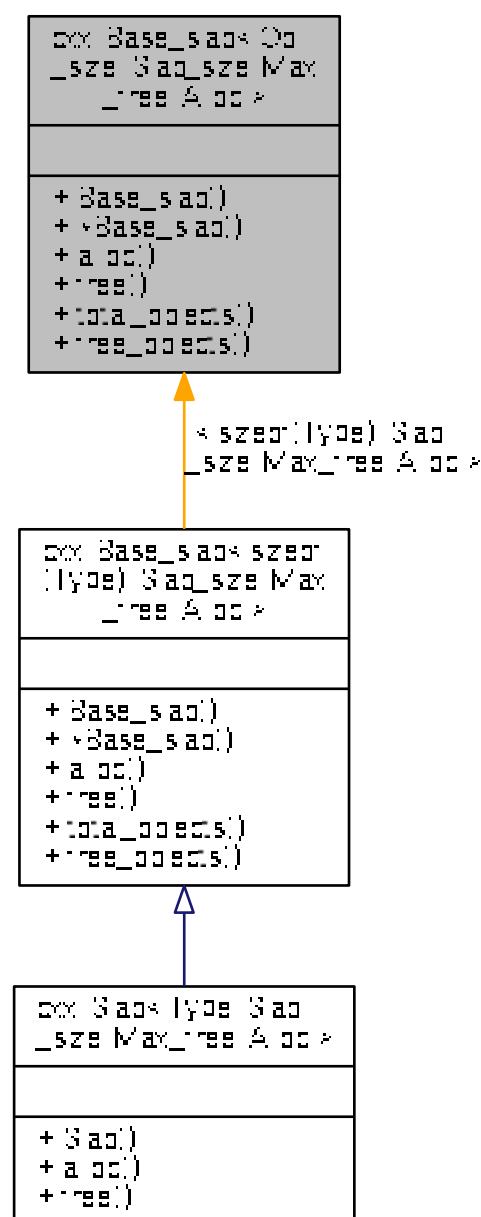
Inheritance diagram for cxx::Avl_tree_node:



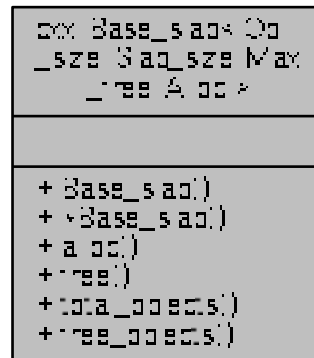
14.6 `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >` Class Template Reference

Basic slab allocator.

Inheritance diagram for `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`:



Collaboration diagram for `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`:



Public Types

- enum { [object_size](#) = Obj_size, [slab_size](#) = Slab_size, [objects_per_slab](#) = (Slab_size - sizeof(Slab_head)) / object_size, [max_free_slabs](#) = Max_free }
- typedef Alloc< Slab_i > [Slab_alloc](#)

Type of the allocator for the slab caches.

Public Member Functions

- unsigned [total_objects](#) () const throw ()
Get the total number of objects managed by the slab allocator.
- unsigned [free_objects](#) () const throw ()
Get the total number of objects managed by the slab allocator.

14.6.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>
class cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >
```

Basic slab allocator.

Parameters

<i>Obj_size</i>	The size of the objects managed by the allocator (in bytes).
<i>Slab_size</i>	The size of a slab cache (in bytes).
<i>Max_free</i>	The maximum number of free slab caches. When this limit is reached slab caches are freed.
<i>Alloc</i>	The allocator that is used to allocate the slab caches.

Definition at line 40 of file [slab_alloc](#).

14.6.2 Member Enumeration Documentation

14.6.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

Enumerator

<code>object_size</code>	size of an object.
<code>slab_size</code>	size of a slab cache.
<code>objects_per_slab</code>	objects per slab cache.
<code>max_free_slabs</code>	maximum number of free slab caches.

Definition at line 63 of file [slab_alloc](#).

14.6.3 Member Function Documentation

14.6.3.1 `free_objects()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >::free\_objects ( ) const throw
) [inline]
```

Get the total number of objects managed by the slab allocator.

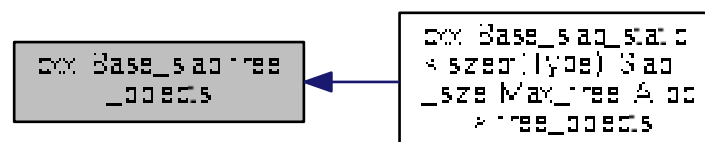
Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 271 of file [slab_alloc](#).

Referenced by [cxx::Base_slab_static< sizeof\(Type\), Slab_size, Max_free, Alloc >::free_objects\(\)](#).

Here is the caller graph for this function:



14.6.3.2 total_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::total_objects ( ) const throw
() [inline]
```

Get the total number of objects managed by the slab allocator.

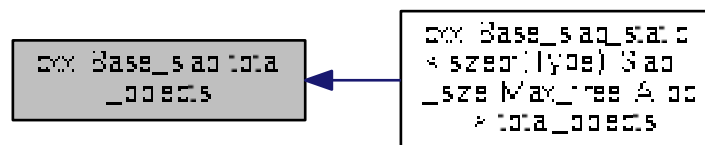
Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 263 of file [slab_alloc](#).

Referenced by [cxx::Base_slab_static< sizeof\(Type\), Slab_size, Max_free, Alloc >::total_objects\(\)](#).

Here is the caller graph for this function:



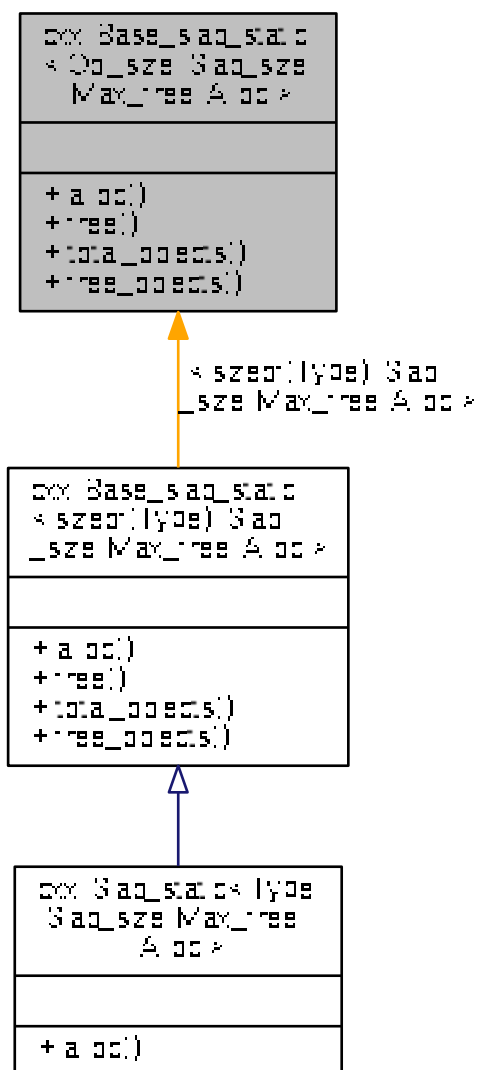
The documentation for this class was generated from the following file:

- `I4/cxx/slab_alloc`

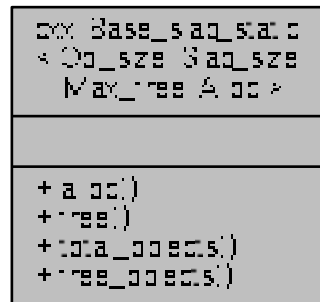
14.7 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

Inheritance diagram for cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >:



Collaboration diagram for `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`:



Public Types

- enum { `object_size` = `Obj_size`, `slab_size` = `Slab_size`, `objects_per_slab` = `_A::objects_per_slab`, `max_free_slabs` = `Max_free` }

Public Member Functions

- void * `alloc` () throw ()
Allocate an object.
- void `free` (void *p) throw ()
Free the given object (p).
- unsigned `total_objects` () const throw ()
Get the total number of objects managed by the slab allocator.
- unsigned `free_objects` () const throw ()
Get the number of free objects in the slab allocator.

14.7.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>
class cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

Parameters

<i>Obj_size</i>	The size of an object managed by the slab allocator.
<i>Slab_size</i>	The size of a slab cache.
<i>Max_free</i>	The maximum number of free slab caches.
<i>Alloc</i>	The allocator for the slab caches.

This slab allocator class is useful for merging slab allocators with the same parameters (equal *Obj_size*, *Slab_size*, *Max_free*, and *Alloc* parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 348 of file `slab_alloc`.

14.7.2 Member Enumeration Documentation

14.7.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

Enumerator

<code>object_size</code>	size of an object.
<code>slab_size</code>	size of a slab cache.
<code>objects_per_slab</code>	number of objects per slab cache.
<code>max_free_slabs</code>	maximum number of free slab caches.

Definition at line 355 of file `slab_alloc`.

14.7.3 Member Function Documentation

14.7.3.1 `alloc()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void* cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::alloc ( ) throw ( ) [inline]
```

Allocate an object.

Definition at line 365 of file `slab_alloc`.

14.7.3.2 `free()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * p ) throw ( ) [inline]
```

Free the given object (*p*).

Parameters

<i>p</i>	The pointer to the object to free.
----------	------------------------------------

Precondition

p must be a pointer to an object allocated by this allocator.

Definition at line 371 of file [slab_alloc](#).

14.7.3.3 free_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::free_objects ( ) const
throw )    [inline]
```

Get the number of free objects in the slab allocator.

Returns

The number of free objects in all free and partially used slab caches managed by this allocator.

Note

The value is the merged value for all equal parameterized [Base_slab_static](#) instances.

Definition at line 389 of file [slab_alloc](#).

14.7.3.4 total_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::total_objects ( )
const throw )    [inline]
```

Get the total number of objects managed by the slab allocator.

Returns

The number of objects managed by the allocator (including the free objects).

Note

The value is the merged value for all equal parameterized [Base_slab_static](#) instances.

Definition at line 380 of file [slab_alloc](#).

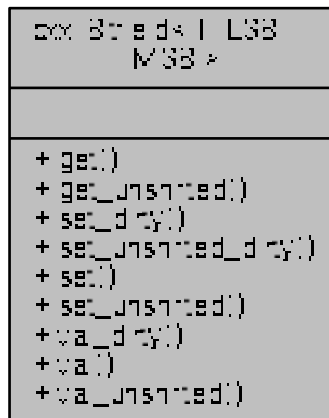
The documentation for this class was generated from the following file:

- [l4/cxx/slab_alloc](#)

14.8 `cxx::Bitfield< T, LSB, MSB >` Class Template Reference

Definition for a member (part) of a bit field.

Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >`:



Data Structures

- class [Value](#)
Internal helper type.
- class [Value_base](#)
Internal helper type.
- class [Value_unshifted](#)
Internal helper type.

Public Types

- enum { [Bits](#) = MSB + 1 - LSB, [Lsb](#) = LSB, [Msb](#) = MSB }
- enum [Masks](#) : T { [Low_mask](#) = ((T)~0ULL) >> (sizeof(T)*8 - Bits), [Mask](#) = Low_mask << Lsb }
- typedef Best_type< [Bits](#) >::Type [Bits_type](#)
Type to hold at least [Bits](#) bits.
- typedef Best_type< [Bits](#)+[Lsb](#) >::Type [Shift_type](#)
Type to hold at least [Bits](#) + [Lsb](#) bits.
- typedef [Value](#)< T & > [Ref](#)
Reference type to access the bits inside a raw bit field.
- typedef [Value](#)< T const > [Val](#)
[Value](#) type to access the bits inside a raw bit field.
- typedef [Value_unshifted](#)< T & > [Ref_unshifted](#)
Reference type to access the bits inside a raw bit field (in place).
- typedef [Value_unshifted](#)< T const > [Val_unshifted](#)
[Value](#) type to access the bits inside a raw bit field (in place).

Static Public Member Functions

- static [Bits_type](#) [get](#) ([Shift_type](#) val)
Get the bits out of val.
- static [T](#) [get_unshifted](#) ([Shift_type](#) val)
Get the bits in place out of val.
- static [T](#) [set_dirty](#) ([T](#) dest, [Shift_type](#) val)
Set the bits corresponding to val.
- static [T](#) [set_unshifted_dirty](#) ([T](#) dest, [Shift_type](#) val)
Set the bits corresponding to val.
- static [T](#) [set](#) ([T](#) dest, [Bits_type](#) val)
Set the bits corresponding to val.
- static [T](#) [set_unshifted](#) ([T](#) dest, [Shift_type](#) val)
Set the bits corresponding to val.
- static [T](#) [val_dirty](#) ([Shift_type](#) val)
Get the shifted bits for val.
- static [T](#) [val](#) ([Bits_type](#) val)
Get the shifted bits for val.
- static [T](#) [val_unshifted](#) ([Shift_type](#) val)
Get the shifted bits for val.

14.8.1 Detailed Description

```
template<typename T, unsigned LSB, unsigned MSB>
class cxx::Bitfield< T, LSB, MSB >
```

Definition for a member (part) of a bit field.

Parameters

<i>T</i>	the underlying type of the bit field.
<i>LSB</i>	the least significant bit of our bits.
<i>MSB</i>	the mos significant bit if our bits.

Definition at line 34 of file [bitfield](#).

14.8.2 Member Typedef Documentation

14.8.2.1 Bits_type

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Best_type<Bits>::Type cxx::Bitfield< T, LSB, MSB >::Bits_type
```

Type to hold at least [Bits](#) bits.

This type can handle all values that can be stored in this part of the bit field.

Definition at line 78 of file [bitfield](#).

14.8.2.2 Ref

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value<T&> cxx::Bitfield< T, LSB, MSB >::Ref
```

Reference type to access the bits inside a raw bit field.

Definition at line 218 of file [bitfield](#).

14.8.2.3 Ref_unshifted

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value_unshifted<T&> cxx::Bitfield< T, LSB, MSB >::Ref_unshifted
```

Reference type to access the bits inside a raw bit field (in place).

Definition at line 223 of file [bitfield](#).

14.8.2.4 Shift_type

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Best_type<Bits + Lsb>::Type cxx::Bitfield< T, LSB, MSB >::Shift_type
```

Type to hold at least `Bits + Lsb` bits.

This type can handle all values that can be stored in this part of the bit field when they are at the target location (`Lsb` bits shifted to the left).

Definition at line 86 of file [bitfield](#).

14.8.2.5 Val

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value<T const> cxx::Bitfield< T, LSB, MSB >::Val
```

`Value` type to access the bits inside a raw bit field.

Definition at line 220 of file [bitfield](#).

14.8.2.6 Val_unshifted

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value_unshifted<T const> cxx::Bitfield< T, LSB, MSB >::Val_unshifted
```

`Value` type to access the bits inside a raw bit field (in place).

Definition at line 225 of file [bitfield](#).

14.8.3 Member Enumeration Documentation

14.8.3.1 anonymous enum

```
template<typename T , unsigned LSB, unsigned MSB>
anonymous enum
```

Enumerator

Bits	Number of bits.
Lsb	index of the LSB
Msb	index of the MSB

Definition at line 58 of file [bitfield](#).

14.8.3.2 Masks

```
template<typename T , unsigned LSB, unsigned MSB>
enum cxx::Bitfield::Masks : T
```

Enumerator

Low_mask	Mask value to get Bits bits.
Mask	Mask value to the bits out of a <i>T</i> .

Definition at line 65 of file [bitfield](#).

14.8.4 Member Function Documentation

14.8.4.1 get()

```
template<typename T , unsigned LSB, unsigned MSB>
static Bits\_type cxx::Bitfield< T, LSB, MSB >::get (
    Shift\_type val ) [inline], [static]
```

Get the bits out of *val*.

Parameters

<i>val</i>	the raw value of the whole bit field.
------------	---------------------------------------

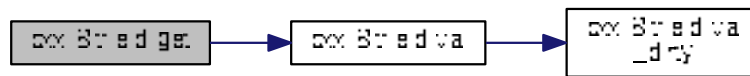
Returns

the bits form [Lsb](#) to [Msb](#) shifted to the right.

Definition at line 100 of file [bitfield](#).

References [cxx::Bitfield](#)< T, LSB, MSB >::Low_mask, [cxx::Bitfield](#)< T, LSB, MSB >::Lsb, and [cxx::Bitfield](#)< T, LSB, MSB >::val().

Here is the call graph for this function:



14.8.4.2 `get_unshifted()`

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::get_unshifted (
    Shift_type val ) [inline], [static]
```

Get the bits in place out of *val*.

Parameters

<i>val</i>	the raw value of the whole bit field.
------------	---------------------------------------

Returns

the bits from `Lsb` to `Msb` (unshifted).

This means other bits are masked out, however the result is not shifted to the right,

Definition at line 110 of file `bitfield`.

References `cxx::Bitfield< T, LSB, MSB >::Mask`.

14.8.4.3 `set()`

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set (
    T dest,
    Bits_type val ) [inline], [static]
```

Set the bits corresponding to *val*.

Parameters

<i>dest</i>	the current value of the whole bit field.
<i>val</i>	the value to set into the bits.

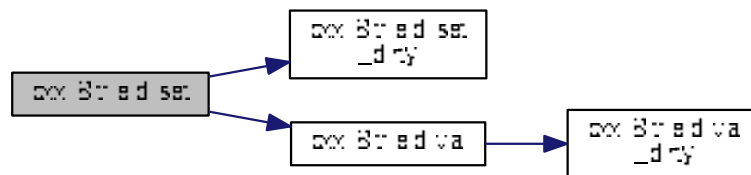
Returns

the new value of the whole bit field.

Definition at line 146 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Low_mask](#), [cxx::Bitfield< T, LSB, MSB >::set_dirty\(\)](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:

**14.8.4.4 set_dirty()**

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_dirty (
    T dest,
    Shift_type val ) [inline], [static]
  
```

Set the bits corresponding to *val*.

Parameters

<i>dest</i>	the current value of the whole bit field.
<i>val</i>	the value to set into the bits.

Returns

the new value of the whole bit field.

Precondition

val must contain not more than bits than [Bits](#).

Note

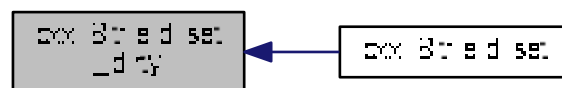
This function does not mask *val* to the right number of bits.

Definition at line 120 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Lsb](#), and [cxx::Bitfield< T, LSB, MSB >::Mask](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::set\(\)](#).

Here is the caller graph for this function:

**14.8.4.5 `set_unshifted()`**

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set\_unshifted (
    T dest,
    Shift\_type val ) [inline], [static]
  
```

Set the bits corresponding to *val*.

Parameters

<i>dest</i>	the current value of the whole bit field.
<i>val</i>	the value shifted Lsb bits to the left that shall be set into the bits.

Returns

the new value of the whole bit field.

Definition at line 155 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty\(\)](#).

Here is the call graph for this function:



14.8.4.6 set_unshifted_dirty()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty (
    T dest,
    Shift_type val ) [inline], [static]
```

Set the bits corresponding to *val*.

Parameters

<i>dest</i>	the current value of the whole bit field.
<i>val</i>	the value shifted Lsb bits to the left that shall be set into the bits.

Returns

the new value of the whole bit field.

Precondition

val must contain not more than bits than [Bits](#) shifted [Lsb](#) bits to the left.

Note

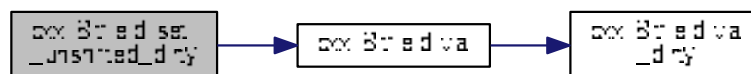
This function does not mask *val* to the right number of bits.

Definition at line [135](#) of file [bitfield](#).

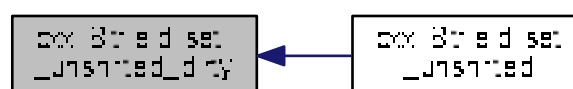
References [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::set_unshifted\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.8.4.7 `val()`

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val (
    Bits_type val ) [inline], [static]
```

Get the shifted bits for *val*.

Parameters

<i>val</i>	the value to set into the bits.
------------	---------------------------------

Returns

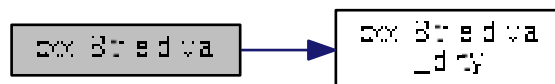
the raw bit field value containing.

Definition at line 170 of file `bitfield`.

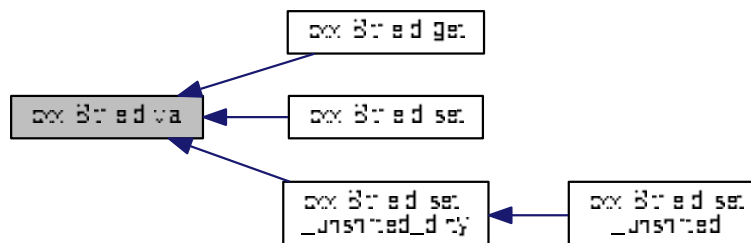
References `cxx::Bitfield< T, LSB, MSB >::Low_mask`, and `cxx::Bitfield< T, LSB, MSB >::val_dirty`.

Referenced by `cxx::Bitfield< T, LSB, MSB >::get()`, `cxx::Bitfield< T, LSB, MSB >::set()`, and `cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty()`.

Here is the call graph for this function:



Here is the caller graph for this function:



Parameters

<i>val</i>	the value shifted Lsb bits to the left that shall be set into the bits.
------------	---

Returns

the raw bit field value containing.

Definition at line 177 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Mask](#).

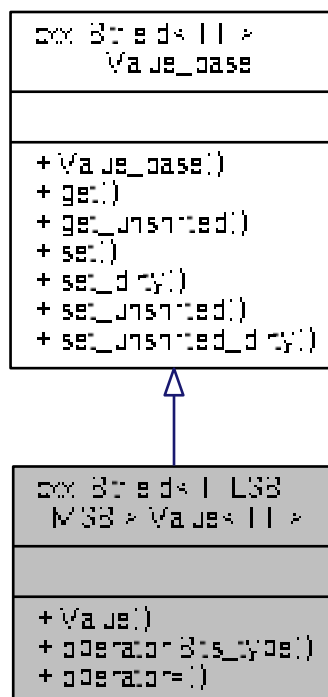
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

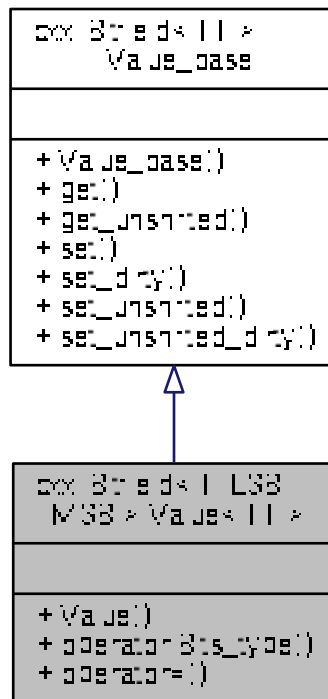
14.9 `cxx::Bitfield< T, LSB, MSB >::Value< TT >` Class Template Reference

Internal helper type.

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



14.9.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value< TT >

```

Internal helper type.

Definition at line 199 of file [bitfield](#).

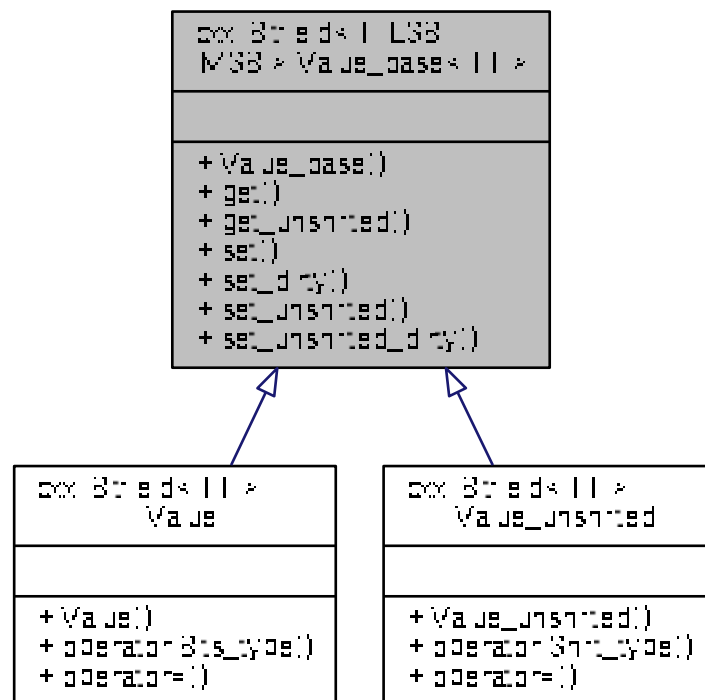
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

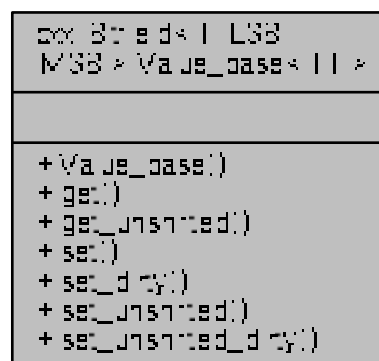
14.10 `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >` Class Template Reference

Internal helper type.

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



14.10.1 Detailed Description

```
template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_base< TT >
```

Internal helper type.

Definition at line 181 of file [bitfield](#).

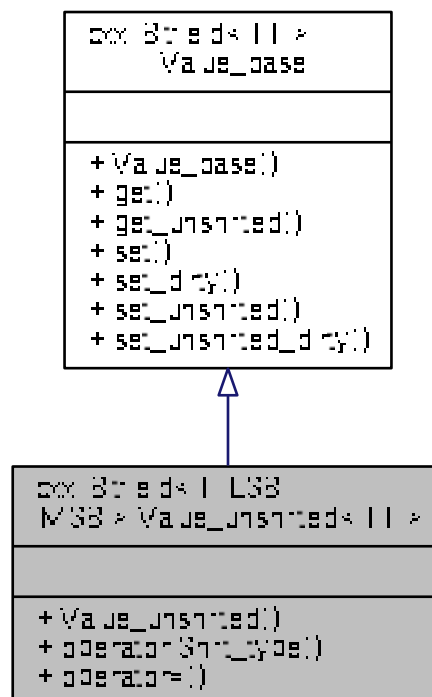
The documentation for this class was generated from the following file:

- I4/cxx/bitfield

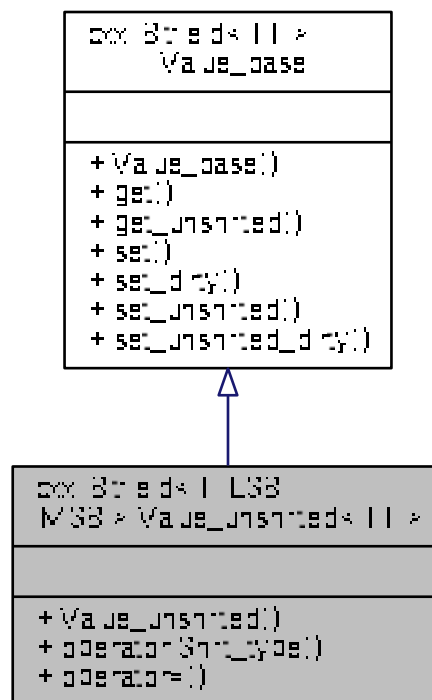
14.11 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



14.11.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >
  
```

Internal helper type.

Definition at line 209 of file [bitfield](#).

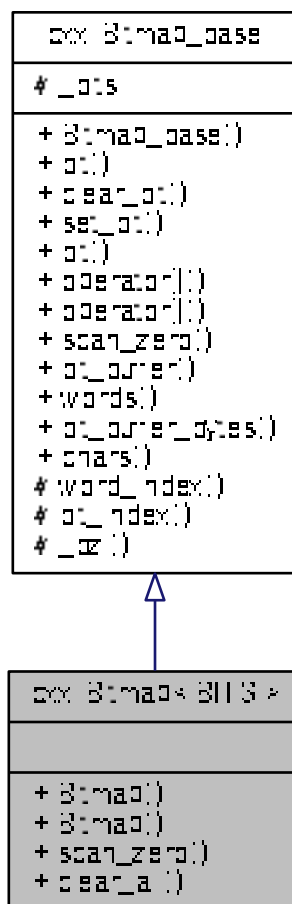
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

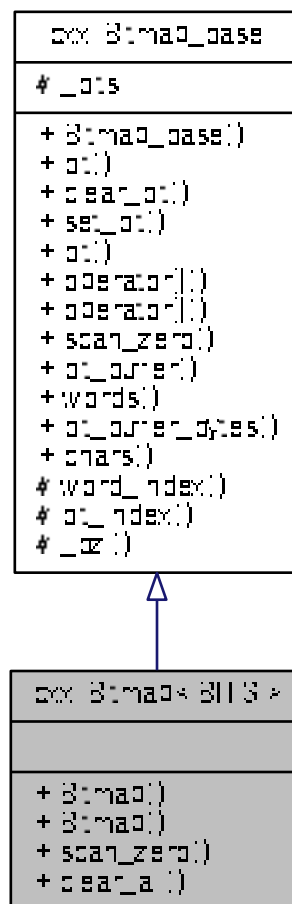
14.12 `cxx::Bitmap< BITS >` Class Template Reference

A static bit map.

Inheritance diagram for `cxx::Bitmap< BITS >`:



Collaboration diagram for cxx::Bitmap< BITS >:



Public Member Functions

- `Bitmap ()` throw ()
Create a bitmap with BITS bits.
- long `scan_zero` (long start_bit=0) const throw ()
Scan for the first zero bit.

Additional Inherited Members

14.12.1 Detailed Description

```
template<int BITS>
class cxx::Bitmap< BITS >
```

A static bit map.

Parameters

<i>BITS</i>	the number of bits that shall be in the bitmap.
-------------	---

Definition at line 180 of file [bitmap](#).

14.12.2 Constructor & Destructor Documentation

14.12.2.1 Bitmap()

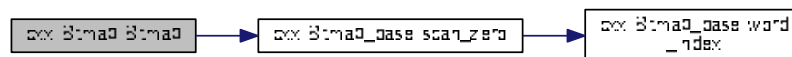
```
template<int BITS>
cxx::Bitmap< BITS >::Bitmap ( ) throw ( ) [inline]
```

Create a bitmap with *BITS* bits.

Definition at line 187 of file [bitmap](#).

References [cxx::Bitmap_base::scan_zero\(\)](#).

Here is the call graph for this function:



14.12.3 Member Function Documentation

14.12.3.1 scan_zero()

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0 ) const throw ( ) [inline]
```

Scan for the first zero bit.

Parameters

<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.
------------------	--

Return values

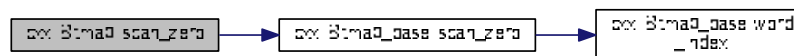
<code>>=</code>	0 Number of first zero bit found.
<code>-1</code>	All bits at <code>start_bit</code> or higher are set.

Compared to [Bitmap_base::scan_zero\(\)](#), the upper bound is set to BITS.

Definition at line 285 of file [bitmap](#).

References [cxx::Bitmap_base::scan_zero\(\)](#).

Here is the call graph for this function:



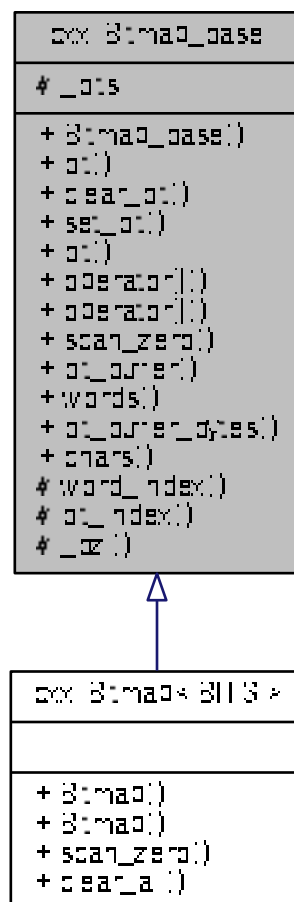
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

14.13 `cxx::Bitmap_base` Class Reference

Basic bitmap abstraction.

Inheritance diagram for `cxx::Bitmap_base`:



Collaboration diagram for cxx::Bitmap_base:

<code>cxx::Bitmap_base</code>
<code># _bits</code>
<code>+ Bitmap_base()</code> <code>+ bit()</code> <code>+ clear_bit()</code> <code>+ set_bit()</code> <code>+ bit()</code> <code>+ operator[]()</code> <code>+ operator[]()</code> <code>+ scan_zero()</code> <code>+ bit_index()</code> <code>+ words()</code> <code>+ bit_index_bytes()</code> <code>+ chars()</code> <code># word_index()</code> <code># bit_index()</code> <code># _w()</code>

Data Structures

- class [Bit](#)
A writeable bit in a bitmap.
- class [Char](#)
Helper abstraction for a byte contained in the bitmap.
- class [Word](#)
Helper abstraction for a word contained in the bitmap.

Public Member Functions

- void [bit](#) (long bit, bool on) throw ()
Set the value of bit bit to on.
- void [clear_bit](#) (long bit) throw ()
Clear bit bit.
- void [set_bit](#) (long bit) throw ()
Set bit bit.
- [word_type](#) [bit](#) (long bit) const throw ()
Get the truth value of a bit.
- [word_type](#) [operator\[\]](#) (long bit) const throw ()
Get the bit at index bit.
- [Bit](#) [operator\[\]](#) (long bit) throw ()
Get the lvalue for the bit at index bit.
- long [scan_zero](#) (long max_bit, long start_bit=0) const throw ()
Scan for the first zero bit.

Static Public Member Functions

- static long [words](#) (long bits) throw ()
Get the number of Words that are used for the bitmap.
- static long [chars](#) (long bits) throw ()
Get the number of chars that are used for the bitmap.

Protected Types

- enum { [W_bits](#) = sizeof(word_type) * 8, [C_bits](#) = 8 }
- typedef unsigned long [word_type](#)
Data type for each element of the bit buffer.

Static Protected Member Functions

- static unsigned [word_index](#) (unsigned [bit](#))
Get the word index for the given bit.
- static unsigned [bit_index](#) (unsigned [bit](#))
Get the bit index within word_type for the given bit.

Protected Attributes

- [word_type](#) * [_bits](#)
Pointer to the buffer storing the bits.

14.13.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 30 of file [bitmap](#).

14.13.2 Member Enumeration Documentation

14.13.2.1 anonymous enum

```
anonymous enum [protected]
```

Enumerator

W_bits	number of bits in word_type
C_bits	number of bits in char

Definition at line 38 of file [bitmap](#).

14.13.3 Member Function Documentation

14.13.3.1 `bit()` [1/2]

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on ) throw ()    [inline]
```

Set the value of bit *bit* to *on*.

Parameters

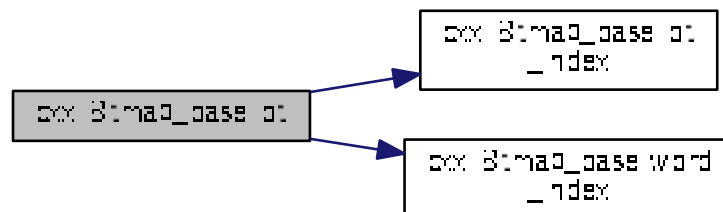
<i>bit</i>	the number of the bit
<i>on</i>	the boolean value that shall be assigned to the bit.

Definition at line 211 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Referenced by [operator\[\]\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.13.3.2 `bit()` [2/2]

```
unsigned long cxx::Bitmap_base::bit (
    long bit ) const throw ()    [inline]
```

Get the truth value of a bit.

Parameters

<i>bit</i>	the number of the bit to read.
------------	--------------------------------

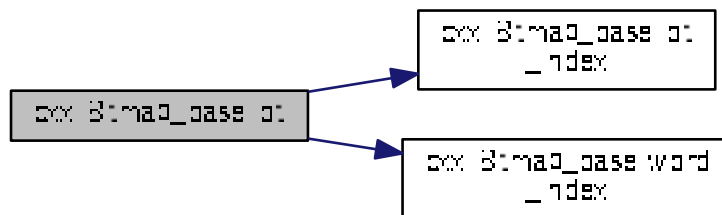
Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 238 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



14.13.3.3 `bit_index()`

```
static unsigned cxx::Bitmap_base::bit_index (
    unsigned bit )    [inline], [static], [protected]
```

Get the bit index within `word_type` for the given bit.

Parameters

<i>bit</i>	the bit index in the bitmap.
------------	------------------------------

Returns

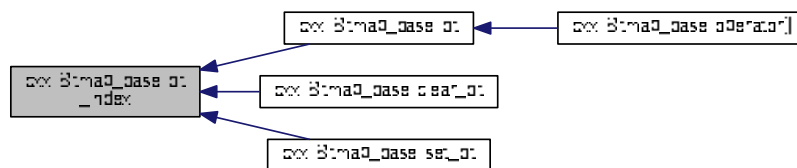
the bit index within word_type (bit % W_bits).

Definition at line 61 of file [bitmap](#).

References [W_bits](#).

Referenced by [bit\(\)](#), [clear_bit\(\)](#), and [set_bit\(\)](#).

Here is the caller graph for this function:

**14.13.3.4 chars()**

```
static long cxx::Bitmap_base::chars (
    long bits ) throw () [inline], [static]
```

Get the number of chars that are used for the bitmap.

Definition at line 98 of file [bitmap](#).

References [C_bits](#).

14.13.3.5 clear_bit()

```
void cxx::Bitmap_base::clear_bit (
    long bit ) throw () [inline]
```

Clear bit *bit*.

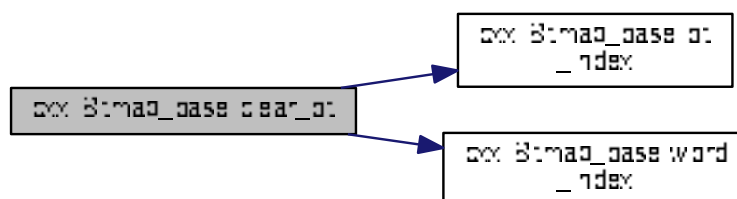
Parameters

<i>bit</i>	the number of the bit to clear.
------------	---------------------------------

Definition at line 220 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



14.13.3.6 operator[]() [1/2]

```
word_type cxx::Bitmap_base::operator[] (
    long bit ) const throw () [inline]
```

Get the bit at index *bit*.

Parameters

<i>bit</i>	the number of the bit to read.
------------	--------------------------------

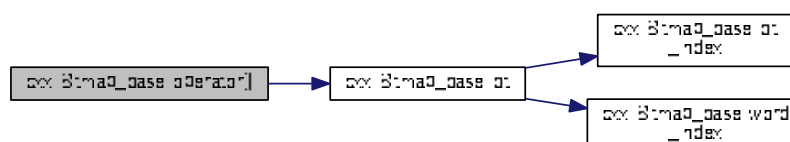
Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 143 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



14.13.3.7 operator[]() [2/2]

```
Bit cxx::Bitmap_base::operator[] (
    long bit ) throw () [inline]
```

Get the lvalue for the bit at index *bit*.

Parameters

<i>bit</i>	the number.
------------	-------------

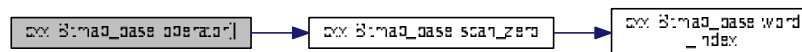
Returns

lvalue for *bit*

Definition at line 151 of file [bitmap](#).

References [_bits](#), and [scan_zero\(\)](#).

Here is the call graph for this function:



14.13.3.8 scan_zero()

```
long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0 ) const throw () [inline]
```

Scan for the first zero bit.

Parameters

<i>max_bit</i>	Upper bound (exclusive) for the scanning operation.
<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.

Return values

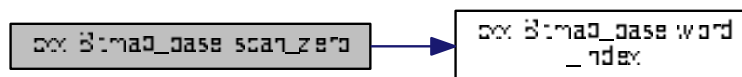
<i>>=</i>	0 Number of first zero bit found.
<i>-1</i>	All bits between <i>start_bit</i> and <i>max_bit</i> are set.

Definition at line 259 of file [bitmap](#).

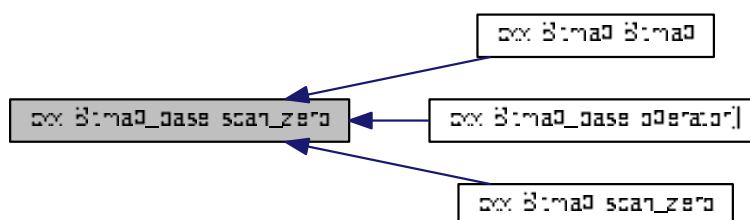
References [W_bits](#), and [word_index\(\)](#).

Referenced by [cxx::Bitmap< BITS >::Bitmap\(\)](#), [operator\[\]\(\)](#), and [cxx::Bitmap< BITS >::scan_zero\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.13.3.9 set_bit()

```
void cxx::Bitmap_base::set_bit (
    long bit ) throw () [inline]
```

Set bit *bit*.

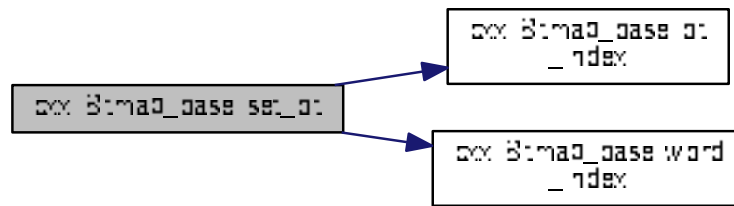
Parameters

<i>bit</i>	the number of the bit to set,
------------	-------------------------------

Definition at line 229 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



14.13.3.10 word_index()

```
static unsigned cxx::Bitmap_base::word_index (
    unsigned bit ) [inline], [static], [protected]
```

Get the word index for the given bit.

Parameters

<i>bit</i>	the index of the bit in question.
------------	-----------------------------------

Returns

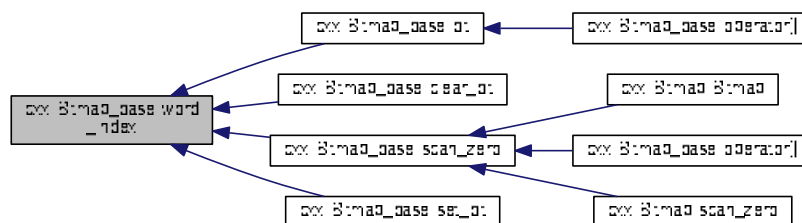
the index in [Bitmap_base::_bits](#) for the given bit (bit / W_bits).

Definition at line 54 of file [bitmap](#).

References [W_bits](#).

Referenced by [bit\(\)](#), [clear_bit\(\)](#), [scan_zero\(\)](#), and [set_bit\(\)](#).

Here is the caller graph for this function:



14.13.3.11 words()

```
static long cxx::Bitmap_base::words (
    long bits ) throw ()    [inline], [static]
```

Get the number of *Words* that are used for the bitmap.

Definition at line 81 of file [bitmap](#).

References [W_bits](#).

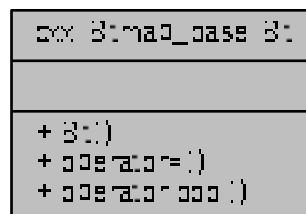
The documentation for this class was generated from the following file:

- I4/cxx/bitmap

14.14 cxx::Bitmap_base::Bit Class Reference

A writeable bit in a bitmap.

Collaboration diagram for cxx::Bitmap_base::Bit:



14.14.1 Detailed Description

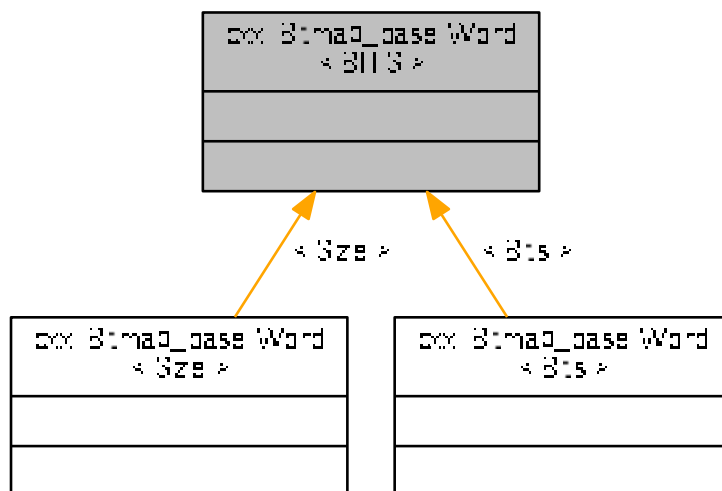
A writeable bit in a bitmap.

Definition at line 66 of file [bitmap](#).

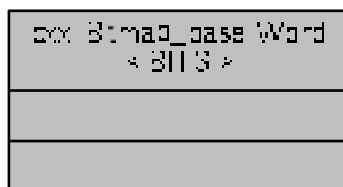
The documentation for this class was generated from the following file:

- I4/cxx/bitmap

Inheritance diagram for `cxx::Bitmap_base::Word< BITS >`:



Collaboration diagram for `cxx::Bitmap_base::Word< BITS >`:



14.16.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Word< BITS >
```

Helper abstraction for a word contained in the bitmap.

Definition at line [87](#) of file [bitmap](#).

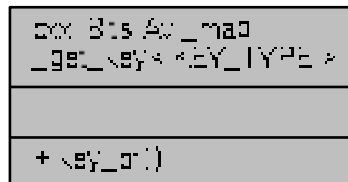
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

14.17 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference

Key-getter for [Avl_map](#).

Collaboration diagram for cxx::Bits::Avl_map_get_key< KEY_TYPE >:



14.17.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_map_get_key< KEY_TYPE >
```

Key-getter for [Avl_map](#).

Definition at line 36 of file [avl_map](#).

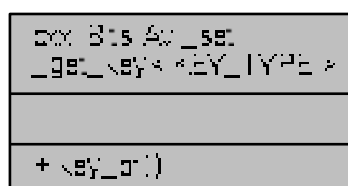
The documentation for this struct was generated from the following file:

- [l4/cxx/avl_map](#)

14.18 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference

Internal, key-getter for [Avl_set](#) nodes.

Collaboration diagram for cxx::Bits::Avl_set_get_key< KEY_TYPE >:



14.18.1 Detailed Description

```
template<typename KEY_TYPE>  
struct cxx::Bits::Avl_set_get_key< KEY_TYPE >
```

Internal, key-getter for [Avl_set](#) nodes.

Definition at line 93 of file [avl_set](#).

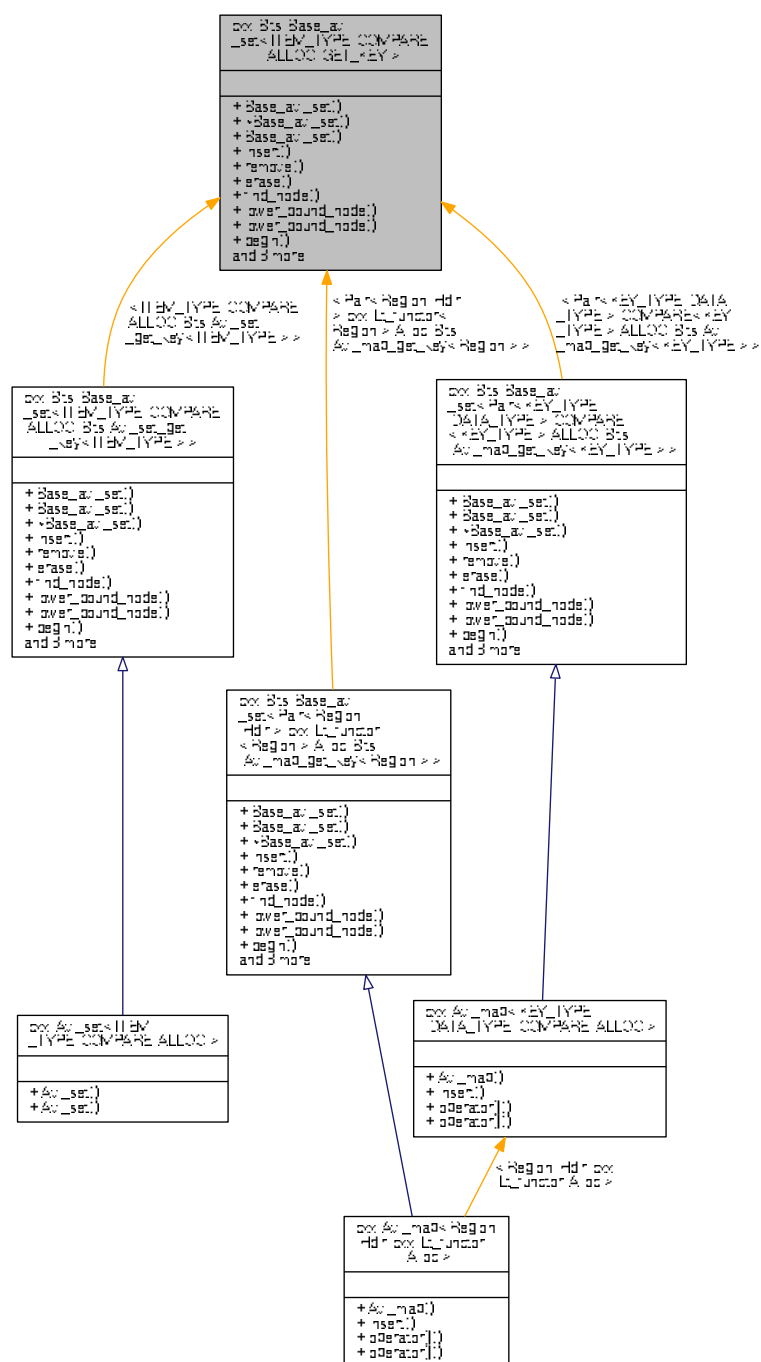
The documentation for this struct was generated from the following file:

- [I4/cxx/avl_set](#)

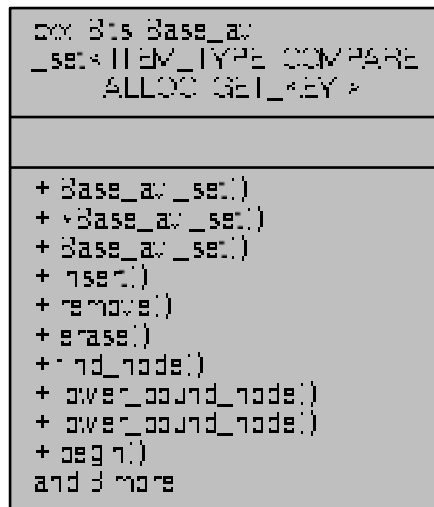
14.19 `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >` Class Template Reference

Internal: AVL set with internally managed nodes.

Inheritance diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`:



Collaboration diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`:



Data Structures

- class [Node](#)
A smart pointer to a tree item.

Public Types

- enum { [E_noent](#) = 2, [E_exist](#) = 17, [E_nomem](#) = 12, [E_inval](#) = 22 }
Return status constants.
- typedef ITEM_TYPE [Item_type](#)
Type for the items store in the set.
- typedef GET_KEY [Get_key](#)
Key-getter type to derive the sort key of an internal node.
- typedef GET_KEY::Key_type [Key_type](#)
Type of the sort key used for the items.
- typedef Type_traits< [Item_type](#) >::Const_type [Const_item_type](#)
Type used for const items within the set.
- typedef COMPARE [Item_compare](#)
Type for the comparison functor.
- typedef ALLOC< _Node > [Node_allocator](#)
Type for the node allocator.
- typedef Avl_set_iter< _Node, [Item_type](#), Fwd > [Iterator](#)
Forward iterator for the set.
- typedef Avl_set_iter< _Node, [Const_item_type](#), Fwd > [Const_iterator](#)
Constant forward iterator for the set.
- typedef Avl_set_iter< _Node, [Item_type](#), Rev > [Rev_iterator](#)
Backward iterator for the set.
- typedef Avl_set_iter< _Node, [Const_item_type](#), Rev > [Const_rev_iterator](#)
Constant backward iterator for the set.

Public Member Functions

- `Base_avl_set (Node_allocator const &alloc=Node_allocator())`
Create a AVL-tree based set.
- `Base_avl_set (Base_avl_set const &o)`
Create a copy of an AVL-tree based set.
- `cxx::Pair< Iterator, int > insert (Item_type const &item)`
Insert an item into the set.
- `int remove (Key_type const &item)`
Remove an item from the set.
- `int erase (Key_type const &item)`
Erase the item with the given key.
- `Node find_node (Key_type const &item) const`
Lookup a node equal to `item`.
- `Node lower_bound_node (Key_type const &key) const`
Find the first node greater or equal to `key`.
- `Const_iterator begin () const`
Get the constant forward iterator for the first element in the set.
- `Const_iterator end () const`
Get the end marker for the constant forward iterator.
- `Iterator begin ()`
Get the mutable forward iterator for the first element of the set.
- `Iterator end ()`
Get the end marker for the mutable forward iterator.
- `Const_rev_iterator rbegin () const`
Get the constant backward iterator for the last element in the set.
- `Const_rev_iterator rend () const`
Get the end marker for the constant backward iterator.
- `Rev_iterator rbegin ()`
Get the mutable backward iterator for the last element of the set.
- `Rev_iterator rend ()`
Get the end marker for the mutable backward iterator.

14.19.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >
```

Internal: AVL set with internally managed nodes.

Use [Avl_set](#), [Avl_map](#), or [Avl_tree](#) in applications.

Template Parameters

<code>ITEM_TYPE</code>	The type of the items to be stored in the set.
<code>COMPARE</code>	The relation to define the partial order, default is to use operator '<'.
<code>ALLOC</code>	The allocator to use for the nodes of the AVL set.
<code>GET_KEY</code>	Sort-key getter (must provide the <code>Key_type</code> and sort-key for an item (of <code>ITEM_TYPE</code>)).

Definition at line 117 of file [avl_set](#).

14.19.2 Member Enumeration Documentation

14.19.2.1 anonymous enum

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_KEY>
anonymous enum
```

Return status constants.

These constants are compatible with the [L4](#) error codes, see [l4_error_code_t](#).

Enumerator

E_noent	Item does not exist.
E_exist	Item exists already.
E_nomem	Memory allocation failed.
E_inval	Internal error.

Definition at line 126 of file [avl_set](#).

14.19.3 Constructor & Destructor Documentation

14.19.3.1 Base_avl_set() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_KEY>
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set (
    Node_allocator const & alloc = Node_allocator() ) [inline], [explicit]
```

Create a AVL-tree based set.

Parameters

<i>alloc</i>	Node allocator.
--------------	---------------------------------

Create an empty set (AVL-tree based).

Definition at line 235 of file [avl_set](#).

14.19.3.2 `Base_avl_set()` [2/2]

```
template<typename Item , class Compare , template< typename A > class Alloc, typename KEY_T↵
YPE >
cxx::Bits::Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Base_avl_set (
    Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > const & o ) [inline]
```

Create a copy of an AVL-tree based set.

Parameters

<i>o</i>	The set to copy.
----------	------------------

Creates a deep copy of the set with all its items.

Definition at line 387 of file `avl_set`.

14.19.4 Member Function Documentation

14.19.4.1 `begin()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_KEY>
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin ( ) const
[inline]
```

Get the constant forward iterator for the first element in the set.

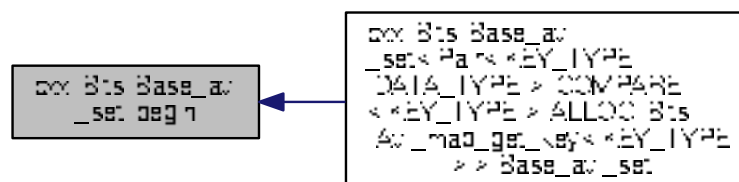
Returns

Constant forward iterator for the first element in the set.

Definition at line 337 of file `avl_set`.

Referenced by `cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC,`
`Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set()`.

Here is the caller graph for this function:



14.19.4.2 `begin()` [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↔
T_KEY>
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin ( ) [inline]
```

Get the mutable forward iterator for the first element of the set.

Returns

The mutable forward iterator for the first element of the set.

Definition at line 348 of file [avl_set](#).

14.19.4.3 `end()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↔
T_KEY>
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end ( ) const
[inline]
```

Get the end marker for the constant forward iterator.

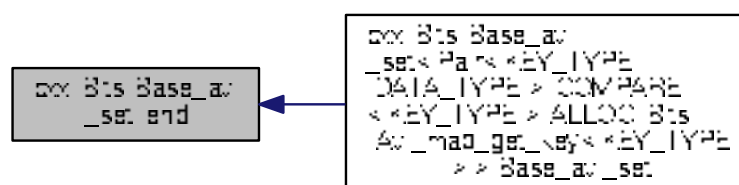
Returns

The end marker for the constant forward iterator.

Definition at line 342 of file [avl_set](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set\(\)](#).

Here is the caller graph for this function:



14.19.4.4 end() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_KEY>
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end ( ) [inline]
```

Get the end marker for the mutable forward iterator.

Returns

The end marker for mutable forward iterator.

Definition at line 353 of file [avl_set](#).

14.19.4.5 erase()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_KEY>
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::erase (
    Key_type const & item ) [inline]
```

Erase the item with the given key.

Parameters

<i>item</i>	The key of the item to remove.
-------------	--------------------------------

Definition at line 305 of file [avl_set](#).

14.19.4.6 find_node()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_KEY>
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::find_node (
    Key_type const & item ) const [inline]
```

Lookup a node equal to *item*.

Parameters

<i>item</i>	The value to search for.
-------------	--------------------------

Returns

A smart pointer to the element found. If no element was found the smart pointer will be invalid.

Definition at line 316 of file [avl_set](#).

14.19.4.7 insert()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↔
T_KEY>
Pair< typename Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Iterator, int > cxx::Bits::↔
Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::insert (
    Item_type const & item )
```

Insert an item into the set.

Parameters

<i>item</i>	The item to insert.
-------------	---------------------

Returns

A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `-E_exist` if the element was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `-E_nomem` when memory for the node could not be allocated. *first* is then invalid.

Insert a new item into the set, each item can only be once in the set.

Definition at line 397 of file [avl_set](#).

14.19.4.8 lower_bound_node()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↔
T_KEY>
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::lower_bound_node (
    Key_type const & key ) const [inline]
```

Find the first node greater or equal to *key*.

Parameters

<i>key</i>	Minimum key to look for.
------------	--------------------------

Returns

Smart pointer to the first node greater or equal to *key*. Will be invalid if no such element was found.

Definition at line 327 of file [avl_set](#).

14.19.4.9 `rbegin()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵  
T_KEY>  
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ( )  
const [inline]
```

Get the constant backward iterator for the last element in the set.

Returns

The constant backward iterator for the last element in the set.

Definition at line 359 of file `avl_set`.

14.19.4.10 `rbegin()` [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵  
T_KEY>  
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ( ) [inline]
```

Get the mutable backward iterator for the last element of the set.

Returns

The mutable backward iterator for the last element of the set.

Definition at line 370 of file `avl_set`.

14.19.4.11 `remove()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵  
T_KEY>  
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::remove (   
    Key_type const & item ) [inline]
```

Remove an item from the set.

Parameters

<i>item</i>	The item to remove.
-------------	---------------------

Return values

0	Success
-E_noent	Item does not exist
-E_inval	Internal error

Definition at line 285 of file [avl_set](#).

14.19.4.12 `rend()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←  
T_KEY>  
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ( )  
const [inline]
```

Get the end marker for the constant backward iterator.

Returns

The end marker for the constant backward iterator.

Definition at line 364 of file [avl_set](#).

14.19.4.13 `rend()` [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←  
T_KEY>  
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ( ) [inline]
```

Get the end marker for the mutable backward iterator.

Returns

The end marker for mutable backward iterator.

Definition at line 375 of file [avl_set](#).

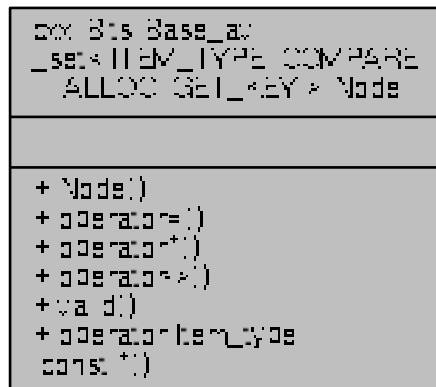
The documentation for this class was generated from the following file:

- [I4/cxx/avl_set](#)

14.20 `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node` Class Reference

A smart pointer to a tree item.

Collaboration diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node`:



Public Member Functions

- [Node](#) ()
Default construction for NIL pointer.
- [Node](#) & [operator=](#) ([Node](#) const &o)
Default assignment.
- [Item_type](#) const & [operator*](#) ()
Dereference the pointer.
- [Item_type](#) const * [operator->](#) ()
Dereferenced member access.
- bool [valid](#) () const
Validity check.
- [operator](#) [Item_type](#) const * ()
Cast to a real item pointer.

14.20.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node
```

A smart pointer to a tree item.

Definition at line 161 of file [avl_set](#).

14.20.2 Member Function Documentation

14.20.2.1 operator*()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_KEY>
Item_type const& cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator*
( ) [inline]
```

Dereference the pointer.

Precondition

[Node](#) is valid.

Definition at line 181 of file [avl_set](#).

14.20.2.2 operator->()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_KEY>
Item_type const* cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator->
( ) [inline]
```

Dereferenced member access.

Precondition

[Node](#) is valid.

Definition at line 187 of file [avl_set](#).

14.20.2.3 valid()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_KEY>
bool cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::valid ( ) const
[inline]
```

Validity check.

Returns

false if the pointer is NIL, true if valid.

Definition at line 193 of file [avl_set](#).

The documentation for this class was generated from the following file:

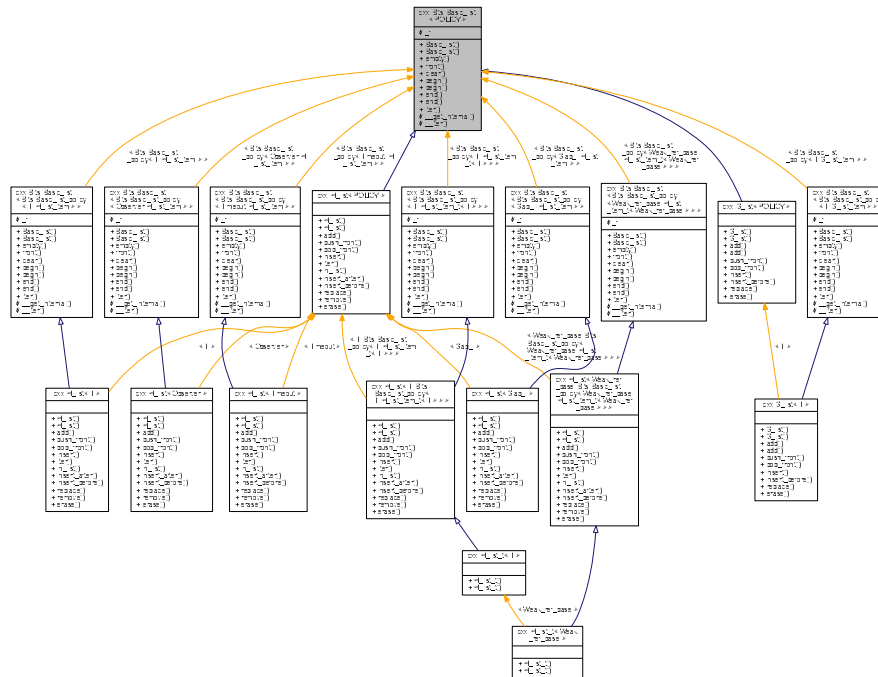
- [I4/cxx/avl_set](#)

14.21 cxx::Bits::Basic_list< POLICY > Class Template Reference

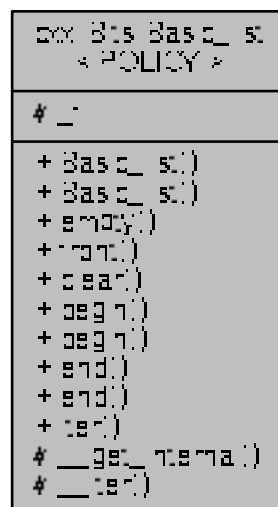
Internal: Common functions for all head-based list implementations.

```
#include <list_basics.h>
```

Inheritance diagram for cxx::Bits::Basic_list< POLICY >:



Collaboration diagram for cxx::Bits::Basic_list< POLICY >:



Public Member Functions

- bool [empty](#) () const
Check if the list is empty.
- Value_type [front](#) () const
Return the first element in the list.
- void [clear](#) ()
Remove all elements from the list.
- Iterator [begin](#) ()
Return an iterator to the beginning of the list.
- Const_iterator [begin](#) () const
Return a const iterator to the beginning of the list.
- Const_iterator [end](#) () const
Return a const iterator to the end of the list.
- Iterator [end](#) ()
Return an iterator to the end of the list.

Static Public Member Functions

- static Const_iterator [iter](#) (Const_value_type c)
Return a const iterator that begins at the given element.

Protected Attributes

- POLICY::Head_type [_f](#)
Pointer to front of the list.

14.21.1 Detailed Description

```
template<typename POLICY>
class cxx::Bits::Basic_list< POLICY >
```

Internal: Common functions for all head-based list implementations.

Definition at line 50 of file [list_basics.h](#).

14.21.2 Member Function Documentation

14.21.2.1 [clear\(\)](#)

```
template<typename POLICY>
void cxx::Bits::Basic_list< POLICY >::clear ( ) [inline]
```

Remove all elements from the list.

After the operation the state of the elements is undefined.

Definition at line 135 of file [list_basics.h](#).

14.21.2.2 iter()

```
template<typename POLICY>
static Const_iterator cxx::Bits::Basic_list< POLICY >::iter (
    Const_value_type c ) [inline], [static]
```

Return a const iterator that begins at the given element.

Parameters

<i>c</i>	Element where the iterator should start.
----------	--

Precondition

The element *c* must already be in a list.

Definition at line 148 of file [list_basics.h](#).

The documentation for this class was generated from the following file:

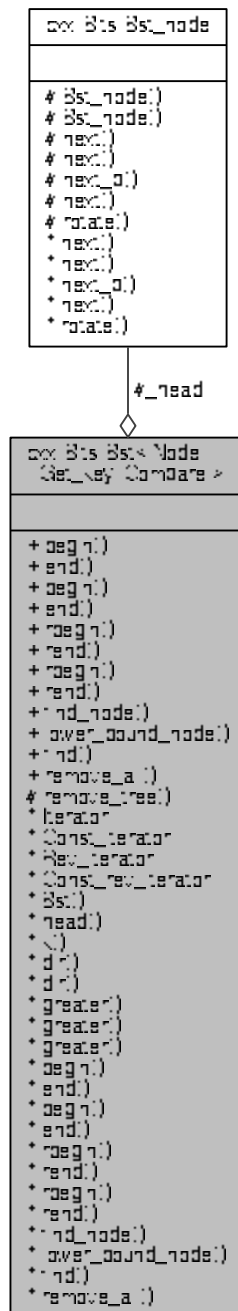
- I4/cxx/bits/list_basics.h

14.22 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference

Basic binary search tree (BST).

```
#include <bst.h>
```


Collaboration diagram for cxx::Bits::Bst< Node, Get_key, Compare >:



Public Types

- typedef Get_key::Key_type [Key_type](#)
The type of key values used to generate the total order of the elements.
- typedef Type_traits< [Key_type](#) >::Param_type [Key_param_type](#)
The type for key parameters.
- typedef Fwd [Fwd_iter_ops](#)

Helper for building forward iterators for different wrapper classes.

- typedef Rev [Rev_iter_ops](#)

Helper for building reverse iterators for different wrapper classes.

Iterators

- typedef __Bst_iter< Node, Node, Fwd > [Iterator](#)
Forward iterator.
- typedef __Bst_iter< Node, Node const, Fwd > [Const_iterator](#)
Constant forward iterator.
- typedef __Bst_iter< Node, Node, Rev > [Rev_iterator](#)
Backward iterator.
- typedef __Bst_iter< Node, Node const, Rev > [Const_rev_iterator](#)
Constant backward.

Public Member Functions

Get default iterators for the ordered tree.

- [Const_iterator begin](#) () const
Get the constant forward iterator for the first element in the set.
- [Const_iterator end](#) () const
Get the end marker for the constant forward iterator.
- [Iterator begin](#) ()
Get the mutable forward iterator for the first element of the set.
- [Iterator end](#) ()
Get the end marker for the mutable forward iterator.
- [Const_rev_iterator rbegin](#) () const
Get the constant backward iterator for the last element in the set.
- [Const_rev_iterator rend](#) () const
Get the end marker for the constant backward iterator.
- [Rev_iterator rbegin](#) ()
Get the mutable backward iterator for the last element of the set.
- [Rev_iterator rend](#) ()
Get the end marker for the mutable backward iterator.

Lookup functions.

- Node * [find_node](#) (Key_param_type key) const
find the node with the given key.
- Node * [lower_bound_node](#) (Key_param_type key) const
find the first node with a key not less than the given key.
- [Const_iterator find](#) (Key_param_type key) const
find the node with the given key.
- template<typename FUNC >
void [remove_all](#) (FUNC &&callback)
Clear the tree.

Static Protected Member Functions

- template<typename FUNC >
static void [remove_tree](#) (Bst_node *head, FUNC &&callback)
Remove all elements in the subtree of head.

Interior access for descendants.

As this class is an intended base class we provide protected access to our interior, use 'using' to make this private in concrete implementations.

- [Bst_node](#) * [_head](#)
The head pointer of the tree.
- [Bst](#) ()
Create an empty tree.
- [Node](#) * [head](#) () const
Access the head node as object of type Node.
- static [Key_type](#) [k](#) ([Bst_node](#) const *n)
Get the key value of n.
- static [Dir](#) [dir](#) ([Key_param_type](#) l, [Key_param_type](#) r)
Get the direction to go from l to search for r.
- static [Dir](#) [dir](#) ([Key_param_type](#) l, [Bst_node](#) const *r)
Get the direction to go from l to search for r.
- static bool [greater](#) ([Key_param_type](#) l, [Key_param_type](#) r)
Is l greater than r.
- static bool [greater](#) ([Key_param_type](#) l, [Bst_node](#) const *r)
Is l greater than r.
- static bool [greater](#) ([Bst_node](#) const *l, [Bst_node](#) const *r)
Is l greater than r.

14.22.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare>
class cxx::Bits::Bst< Node, Get_key, Compare >
```

Basic binary search tree (BST).

This class is intended as a base class for concrete binary search trees, such as an AVL tree. This class already provides the basic lookup methods and iterator definitions for a BST.

Definition at line 40 of file [bst.h](#).

14.22.2 Member Function Documentation

14.22.2.1 `begin()` [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin ( ) const [inline]
```

Get the constant forward iterator for the first element in the set.

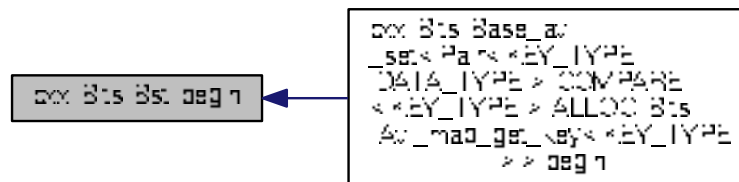
Returns

Constant forward iterator for the first element in the set.

Definition at line 179 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::begin\(\)](#).

Here is the caller graph for this function:

14.22.2.2 `begin()` [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin ( ) [inline]
```

Get the mutable forward iterator for the first element of the set.

Returns

The mutable forward iterator for the first element of the set.

Definition at line 190 of file [bst.h](#).

14.22.2.3 `dir()` [1/2]

```
template<typename Node, typename Get_key, typename Compare>
static Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Key_param_type r ) [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

Parameters

<i>l</i>	is the key to look for.
<i>r</i>	is the key at the current position.

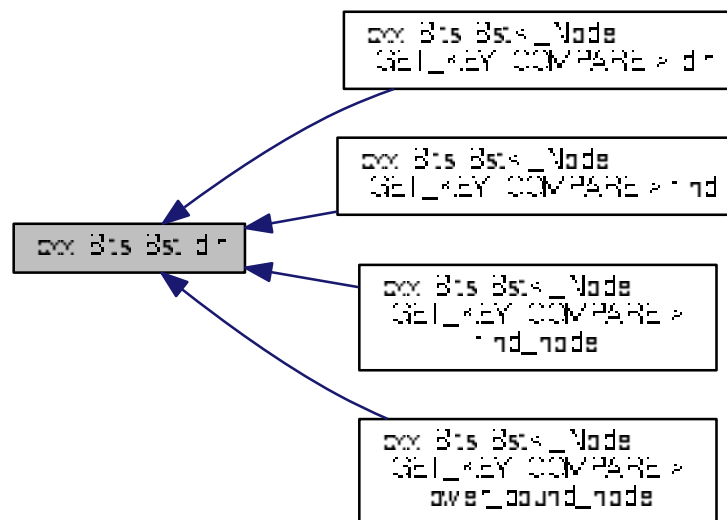
Return values

<i>Direction::L</i>	for left
<i>Direction::R</i>	for right
<i>Direction::N</i>	if <i>l</i> is equal to <i>r</i> .

Definition at line 118 of file [bst.h](#).

Referenced by [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::dir\(\)](#), [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::find\(\)](#), [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::find_node\(\)](#), and [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::lower_bound_node\(\)](#).

Here is the caller graph for this function:

14.22.2.4 `dir()` [2/2]

```

template<typename Node, typename Get_key, typename Compare>
static Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Bst_node const * r ) [inline], [static], [protected]

```

Get the direction to go from *l* to search for *r*.

Parameters

<i>l</i>	is the key to look for.
<i>r</i>	is the node at the current position.

Return values

<i>Direction::L</i>	For left.
<i>Direction::R</i>	For right.
<i>Direction::N</i>	If <i>l</i> is equal to <i>r</i> .

Definition at line 135 of file [bst.h](#).

14.22.2.5 `end()` [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::end ( ) const [inline]
```

Get the end marker for the constant forward iterator.

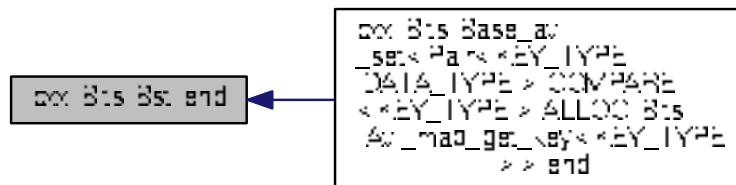
Returns

The end marker for the constant forward iterator.

Definition at line 184 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::end\(\)](#).

Here is the caller graph for this function:



14.22.2.6 end() [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::end ( ) [inline]
```

Get the end marker for the mutable forward iterator.

Returns

The end marker for mutable forward iterator.

Definition at line 195 of file [bst.h](#).

14.22.2.7 find()

```
template<typename Node , typename Get_key , class Compare >
Bst< Node, Get_key, Compare >::Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::find
(
    Key_param_type key ) const [inline]
```

find the node with the given *key*.

Parameters

<i>key</i>	The key value of the element to search.
------------	---

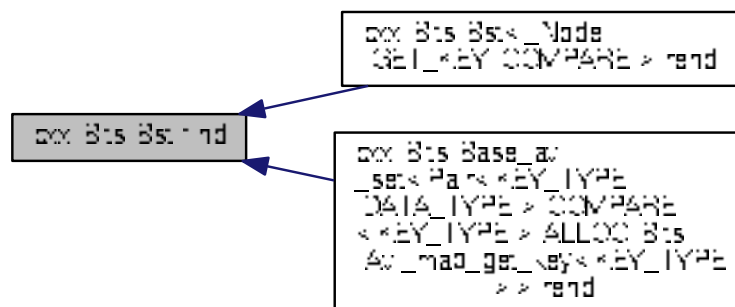
Returns

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 312 of file [bst.h](#).

Referenced by [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend\(\)](#), and [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rend\(\)](#).

Here is the caller graph for this function:



14.22.2.8 find_node()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Bits::Bst< Node, Get_key, Compare >::find_node (
    Key_param_type key ) const [inline]
```

find the node with the given *key*.

Parameters

<i>key</i>	The key value of the element to search.
------------	---

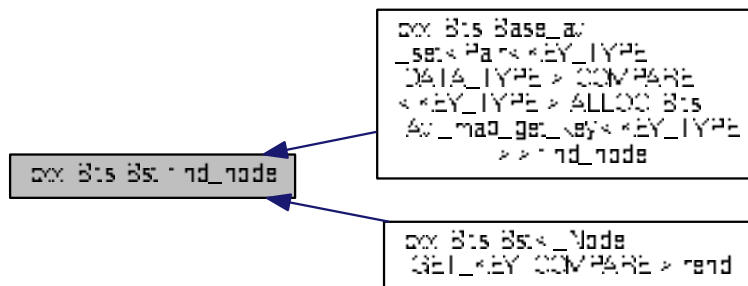
Returns

A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 276 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::find_node\(\)](#), and [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend\(\)](#).

Here is the caller graph for this function:



14.22.2.9 lower_bound_node()

```
template<typename Node , typename Get_key , class Compare >
Node * cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node (
    Key_param_type key ) const [inline]
```

find the first node with a key not less than the given *key*.

Parameters

<i>key</i>	The key value of the element to search.
------------	---

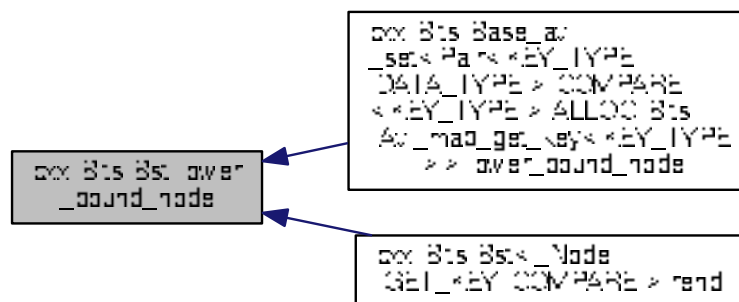
Returns

A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 292 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::lower_bound_node\(\)](#), and [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend\(\)](#).

Here is the caller graph for this function:

14.22.2.10 [rbegin\(\)](#) [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin ( ) const [inline]
```

Get the constant backward iterator for the last element in the set.

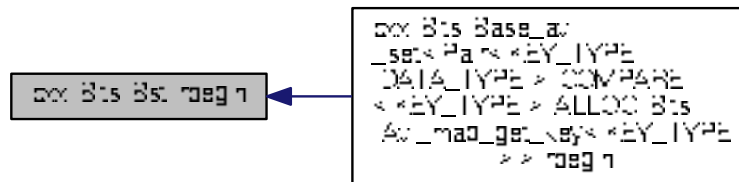
Returns

The constant backward iterator for the last element in the set.

Definition at line 201 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rbegin\(\)](#).

Here is the caller graph for this function:



14.22.2.11 rbeg() [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbeg ( ) [inline]
```

Get the mutable backward iterator for the last element of the set.

Returns

The mutable backward iterator for the last element of the set.

Definition at line 212 of file [bst.h](#).

14.22.2.12 remove_all()

```
template<typename Node, typename Get_key, typename Compare>
template<typename FUNC >
void cxx::Bits::Bst< Node, Get_key, Compare >::remove_all (
    FUNC && callback ) [inline]
```

Clear the tree.

Parameters

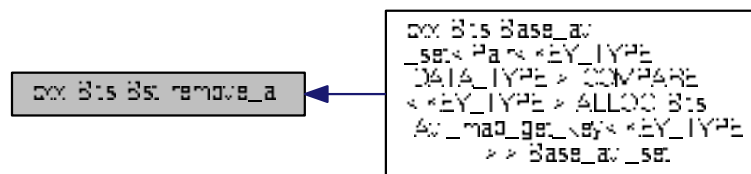
<i>callback</i>	Optional function to be called on each removed element.
-----------------	---

The callback may delete the elements. The function guarantees that the elements are no longer used after the callback has been called.

Definition at line 258 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set\(\)](#).

Here is the caller graph for this function:



14.22.2.13 remove_tree()

```

template<typename Node, typename Get_key, typename Compare>
template<typename FUNC >
static void cxx::Bits::Bst< Node, Get_key, Compare >::remove_tree (
    Bst_node * head,
    FUNC && callback ) [inline], [static], [protected]
  
```

Remove all elements in the subtree of head.

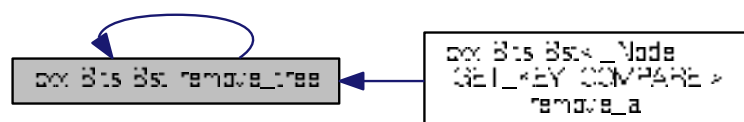
Parameters

<i>head</i>	Head of the the subtree to remove
<i>callback</i>	Optional function called on each removed element.

Definition at line 158 of file [bst.h](#).

Referenced by [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::remove_all\(\)](#), and [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::remove_tree\(\)](#).

Here is the caller graph for this function:



14.22.2.14 `rend()` [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend ( ) const [inline]
```

Get the end marker for the constant backward iterator.

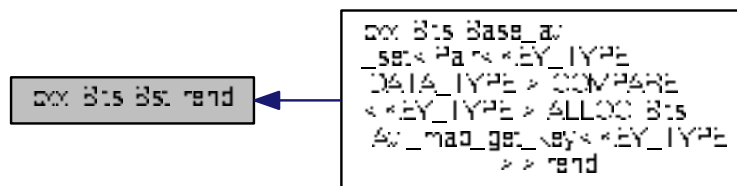
Returns

The end marker for the constant backward iterator.

Definition at line 206 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rend\(\)](#).

Here is the caller graph for this function:

**14.22.2.15** `rend()` [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend ( ) [inline]
```

Get the end marker for the mutable backward iterator.

Returns

The end marker for mutable backward iterator.

Definition at line 217 of file [bst.h](#).

The documentation for this class was generated from the following file:

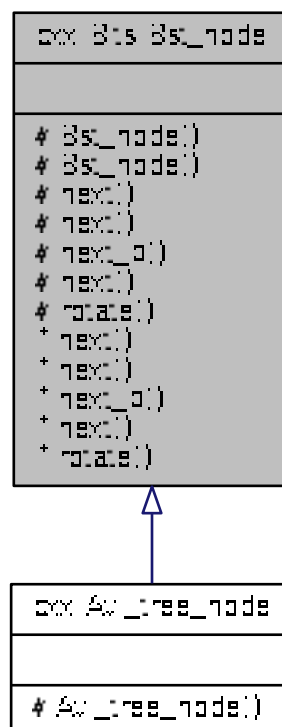
- [l4/cxx/bits/bst.h](#)

14.23 cxx::Bits::Bst_node Class Reference

Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst_node:

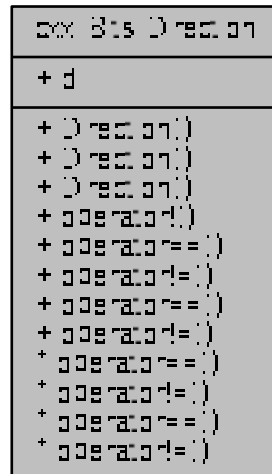


14.24 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

Collaboration diagram for cxx::Bits::Direction:



Public Types

- enum [Direction_e](#) { [L](#) = 0, [R](#) = 1, [N](#) = 2 }

The literal direction values.

Public Member Functions

- [Direction](#) ()
Uninitialized direction.
- [Direction](#) ([Direction_e](#) d)
Convert a literal direction ([L](#), [R](#), [N](#)) to an object.
- [Direction](#) (bool b)
Convert a boolean to a direction (false == [L](#), true == [R](#))
- [Direction operator!](#) () const
Negate the direction.

Comparison operators (equality and inequality)

- bool [operator==](#) ([Direction_e](#) o) const
- bool [operator!=](#) ([Direction_e](#) o) const
- bool [operator==](#) ([Direction](#) o) const
- bool [operator!=](#) ([Direction](#) o) const

14.24.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file [bst_base.h](#).

14.24.2 Member Enumeration Documentation

14.24.2.1 Direction_e

```
enum cxx::Bits::Direction::Direction_e
```

The literal direction values.

Enumerator

L	Go to the left child.
R	Go to the right child.
N	Stop.

Definition at line 42 of file [bst_base.h](#).

14.24.3 Member Function Documentation

14.24.3.1 operator!()

```
Direction cxx::Bits::Direction::operator! ( ) const [inline]
```

Negate the direction.

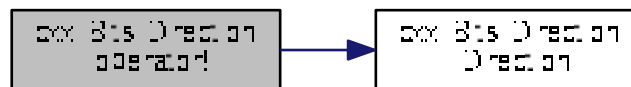
Note

This is only defined for a current value of [L](#) or [R](#)

Definition at line 63 of file [bst_base.h](#).

References [Direction\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

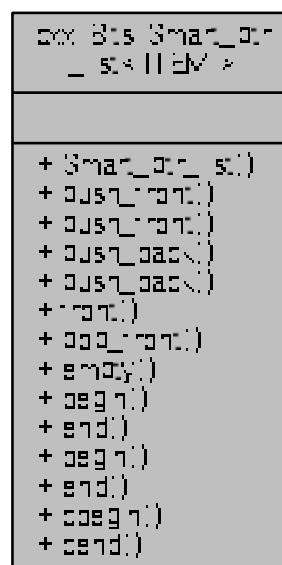
- [l4/cxx/bits/bst_base.h](#)

14.25 `cxx::Bits::Smart_ptr_list< ITEM >` Class Template Reference

[List](#) of smart-pointer-managed objects.

```
#include <smart_ptr_list.h>
```

Collaboration diagram for `cxx::Bits::Smart_ptr_list< ITEM >`:



Public Member Functions

- void [push_front](#) (Next_type &&e)
Add an element to the front of the list.
- void [push_front](#) (Next_type const &e)
Add an element to the front of the list.
- void [push_back](#) (Next_type &&e)
Add an element at the end of the list.
- void [push_back](#) (Next_type const &e)
Add an element at the end of the list.
- Value_type * [front](#) () const
Return a pointer to the first element in the list.
- Next_type [pop_front](#) ()
Remove the element in front of the list and return it.
- bool [empty](#) () const
Check if the list is empty.

14.25.1 Detailed Description

```
template<typename ITEM>
class cxx::Bits::Smart_ptr_list< ITEM >
```

[List](#) of smart-pointer-managed objects.

Template Parameters

<i>ITEM</i>	Type of the list items.
-------------	-------------------------

The list is implemented as a single-linked list connected via smart pointers, so that they are automatically cleaned up when they are removed from the list.

Definition at line [46](#) of file [smart_ptr_list.h](#).

14.25.2 Member Function Documentation

14.25.2.1 pop_front()

```
template<typename ITEM >
Next_type cxx::Bits::Smart_ptr_list< ITEM >::pop_front ( ) [inline]
```

Remove the element in front of the list and return it.

Returns

The element that was previously in front of the list as a managed pointer or a nullptr-equivalent when the list was already empty.

Definition at line [149](#) of file [smart_ptr_list.h](#).

The documentation for this class was generated from the following file:

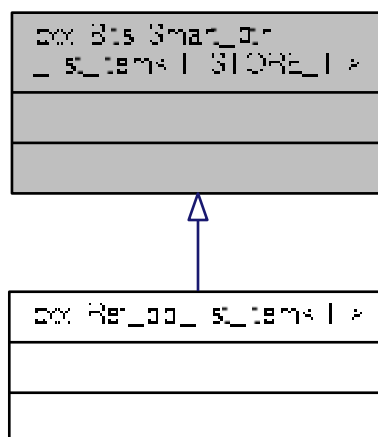
- [I4/cxx/bits/smart_ptr_list.h](#)

14.26 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference

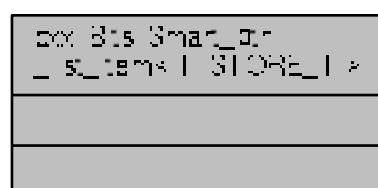
List item for an arbitrary item in a [Smart_ptr_list](#).

```
#include <smart_ptr_list.h>
```

Inheritance diagram for cxx::Bits::Smart_ptr_list_item< T, STORE_T >:



Collaboration diagram for cxx::Bits::Smart_ptr_list_item< T, STORE_T >:



14.26.1 Detailed Description

```
template<typename T, typename STORE_T>
class cxx::Bits::Smart_ptr_list_item< T, STORE_T >
```

List item for an arbitrary item in a [Smart_ptr_list](#).

Template Parameters

<i>T</i>	Type of object to be stored in the list.
<i>STORE↔ _T</i>	Storage type for pointer to next item. The class must implement a <code>get()</code> function that returns a pointer to the stored object and destroy the stored object when the item goes out of scope.

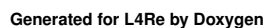
Definition at line 27 of file [smart_ptr_list.h](#).

The documentation for this class was generated from the following file:

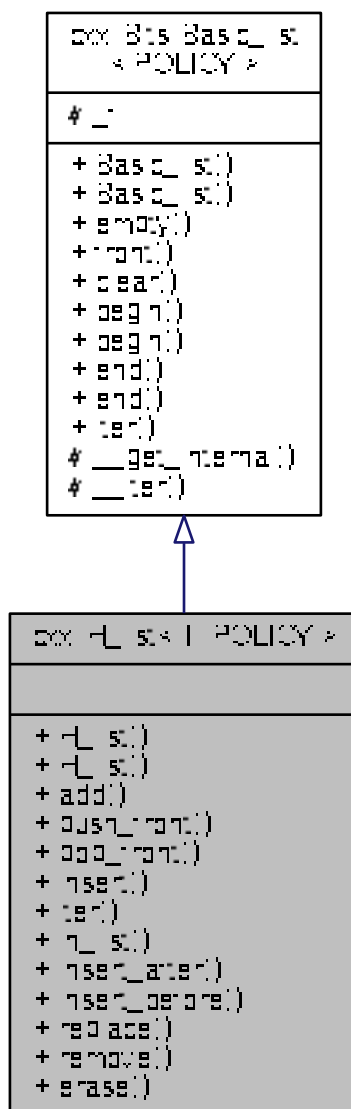
- `I4/cxx/bits/smart_ptr_list.h`

14.27 `cxx::H_list< T, POLICY >` Class Template Reference

General double-linked list of unspecified [cxx::H_list_item](#) elements.



Collaboration diagram for `cxx::H_list< T, POLICY >`:



Public Member Functions

- `void add (T *e)`
Add element to the front of the list.
- `void push_front (T *e)`
Add element to the front of the list.
- `T * pop_front ()`
Remove and return the head element of the list.
- `Iterator insert (T *e, Iterator const &pred)`
Insert an element at the iterator position.

Static Public Member Functions

- static Iterator `iter` (`T *c`)
Return an iterator for an arbitrary list element.
- static bool `in_list` (`T const *e`)
Check if the given element is currently part of a list.
- static Iterator `insert_after` (`T *e`, Iterator const &pred)
Insert an element after the iterator position.
- static void `insert_before` (`T *e`, Iterator const &succ)
Insert an element before the iterator position.
- static void `replace` (`T *p`, `T *e`)
Replace an element in a list with a new element.
- static void `remove` (`T *e`)
Remove the given element from its list.
- static Iterator `erase` (Iterator const &e)
Remove the element at the given iterator position.

Additional Inherited Members

14.27.1 Detailed Description

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
class cxx::H_list< T, POLICY >
```

General double-linked list of unspecified `cxx::H_list_item` elements.

Most of the time, you want to use `H_list_t`.

Definition at line 80 of file `hlist`.

14.27.2 Member Function Documentation

14.27.2.1 `erase()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H_list< T, POLICY >::erase (
    Iterator const & e ) [inline], [static]
```

Remove the element at the given iterator position.

Parameters

<code>e</code>	Iterator pointing to the element to be removed. Must not point to <code>end()</code> .
----------------	--

Returns

New iterator pointing to the element after the removed one.

Note

The `hlist` implementation guarantees that the original iterator is still valid after the element has been removed. In fact, the iterator returned is the same as the one supplied in the `e` parameter.

Definition at line 247 of file [hlist](#).

14.27.2.2 insert()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H\_list< T, POLICY >::insert (
    T * e,
    Iterator const & pred ) [inline]
```

Insert an element at the iterator position.

Parameters

<i>e</i>	New Element to be inserted
<i>pred</i>	Iterator pointing to the element after which the element will be inserted. If end() is given, the element will be inserted at the beginning of the queue.

Returns

Iterator pointing to the newly inserted element.

Definition at line 144 of file [hlist](#).

14.27.2.3 insert_after()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H\_list< T, POLICY >::insert_after (
    T * e,
    Iterator const & pred ) [inline], [static]
```

Insert an element after the iterator position.

Parameters

<i>e</i>	New element to be inserted.
<i>pred</i>	Iterator pointing to the element after which the element will be inserted. Must not be end() .

Returns

Iterator pointing to the newly inserted element.

Precondition

The list must not be empty.

Definition at line 171 of file [hlist](#).

14.27.2.4 `insert_before()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::insert_before (
    T * e,
    Iterator const & succ ) [inline], [static]
```

Insert an element before the iterator position.

Parameters

<i>e</i>	New element to be inserted.
<i>succ</i>	Iterator pointing to the element before which the element will be inserted. Must not be end() .

Definition at line 191 of file [hlist](#).

14.27.2.5 `iter()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static Iterator cxx::H_list< T, POLICY >::iter (
    T * c ) [inline], [static]
```

Return an iterator for an arbitrary list element.

Parameters

<i>c</i>	List element to start the iteration.
----------	--

Returns

A mutable forward iterator.

Precondition

The element must be in a list.

Definition at line 104 of file [hlist](#).

14.27.2.6 pop_front()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
T* cxx::H_list< T, POLICY >::pop_front ( ) [inline]
```

Remove and return the head element of the list.

Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 127 of file [hlist](#).

14.27.2.7 remove()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::remove (
    T * e ) [inline], [static]
```

Remove the given element from its list.

Parameters

<i>e</i>	Element to be removed. Must be in a list.
----------	---

Definition at line 231 of file [hlist](#).

14.27.2.8 replace()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static void cxx::H_list< T, POLICY >::replace (
    T * p,
    T * e ) [inline], [static]
```

Replace an element in a list with a new element.

Parameters

<i>p</i>	Element in list to be replaced.
<i>e</i>	Replacement element, must not yet be in a list.

Precondition

p and *e* must not be NULL.

After the operation the *p* element is no longer in the list and may be reused.

Definition at line 215 of file [hlist](#).

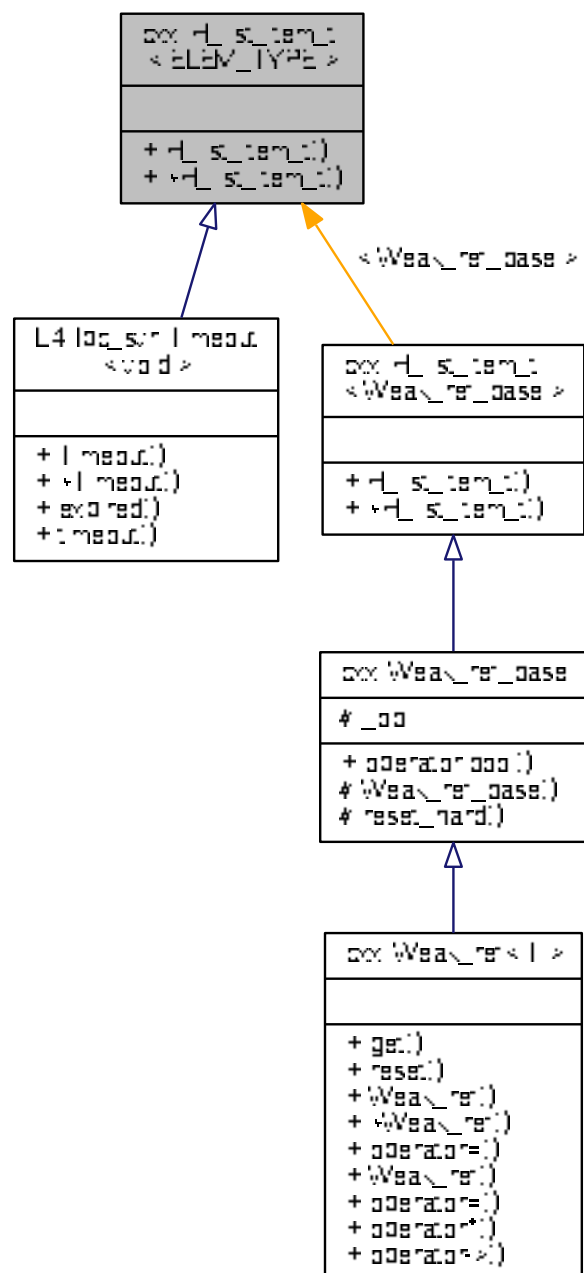
The documentation for this class was generated from the following file:

- l4/cxx/hlist

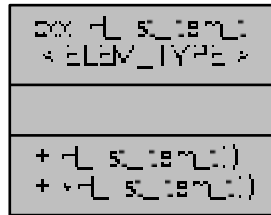
14.28 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference

Basic element type for a double-linked [H_list](#).

Inheritance diagram for cxx::H_list_item_t< ELEM_TYPE >:



Collaboration diagram for `cxx::H_list_item_t< ELEM_TYPE >`:



Public Member Functions

- [H_list_item_t\(\)](#)
Constructor.
- [~H_list_item_t\(\)](#) noexcept
Destructor.

14.28.1 Detailed Description

```
template<typename ELEM_TYPE>
class cxx::H_list_item_t< ELEM_TYPE >
```

Basic element type for a double-linked [H_list](#).

Template Parameters

<code>ELEM_TYPE</code>	Base class of the list element.
------------------------	---------------------------------

Definition at line [33](#) of file [hlist](#).

14.28.2 Constructor & Destructor Documentation

14.28.2.1 H_list_item_t()

```
template<typename ELEM_TYPE>
cxx::H_list_item_t< ELEM_TYPE >::H_list_item_t ( ) [inline]
```

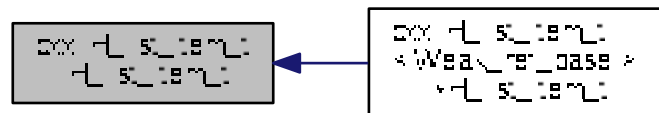
Constructor.

Creates an element that is not in any list.

Definition at line 41 of file [hlist](#).

Referenced by [cxx::H_list_item_t< Weak_ref_base >::~~H_list_item_t\(\)](#).

Here is the caller graph for this function:



14.28.2.2 ~H_list_item_t()

```
template<typename ELEM_TYPE>
cxx::H_list_item_t< ELEM_TYPE >::~~H_list_item_t ( ) [inline], [noexcept]
```

Destructor.

Automatically removes the element from any list it still might be enchainned in.

Definition at line 48 of file [hlist](#).

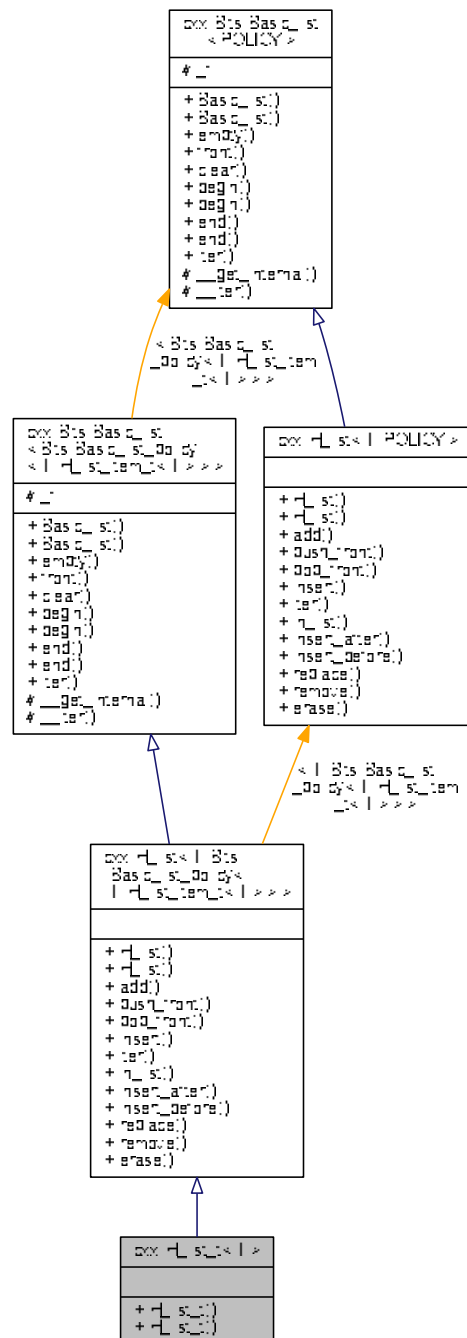
The documentation for this class was generated from the following file:

- [l4/cxx/hlist](#)

14.29 cxx::H_list_t< T > Struct Template Reference

Double-linked list of typed [H_list_item_t](#) elements.

Collaboration diagram for cxx::H_list_t< T >:



Additional Inherited Members

14.29.1 Detailed Description

```
template<typename T>
struct cxx::H_list_t< T >
```

Double-linked list of typed `H_list_item_t` elements.

Note

H_lists are not self-cleaning. Elements that are still chained during destruction are not removed and will therefore be in an undefined state after the destruction.

Definition at line 259 of file [hlist](#).

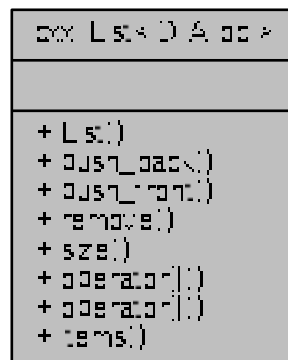
The documentation for this struct was generated from the following file:

- l4/cxx/hlist

14.30 cxx::List< D, Alloc > Class Template Reference

Doubly linked list, with internal allocation.

Collaboration diagram for cxx::List< D, Alloc >:



Data Structures

- class [Iter](#)
Iterator.

Public Member Functions

- void [push_back](#) (D const &d) throw ()
Add element at the end of the list.
- void [push_front](#) (D const &d) throw ()
Add element at the beginning of the list.
- void [remove](#) ([Iter](#) const &i) throw ()
Remove element pointed to by the iterator.
- unsigned long [size](#) () const throw ()
Get the length of the list.
- D const & [operator\[\]](#) (unsigned long idx) const throw ()
Random access.
- D & [operator\[\]](#) (unsigned long idx) throw ()
Random access.
- [Iter](#) [items](#) () throw ()
Get iterator for the list elements.

14.30.1 Detailed Description

```
template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >
```

Doubly linked list, with internal allocation.

Container for items of type D, implemented by a doubly linked list. Alloc defines the allocator policy.

Definition at line 334 of file [list](#).

14.30.2 Member Function Documentation

14.30.2.1 `items()`

```
template<typename D , template< typename A > class Alloc = New_allocator>
Iter cxx::List< D, Alloc >::items ( ) throw )    [inline]
```

Get iterator for the list elements.

Definition at line 412 of file [list](#).

14.30.2.2 `operator[]()` [1/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D const& cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) const throw )    [inline]
```

Random access.

Complexity is O(n).

Definition at line 404 of file [list](#).

14.30.2.3 `operator[]()` [2/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D& cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) throw )    [inline]
```

Random access.

Complexity is O(n).

Definition at line 408 of file [list](#).

14.30.2.4 push_back()

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::push_back (
    D const & d ) throw )    [inline]
```

Add element at the end of the list.

Definition at line 379 of file [list](#).

14.30.2.5 push_front()

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::push_front (
    D const & d ) throw )    [inline]
```

Add element at the beginning of the list.

Definition at line 388 of file [list](#).

14.30.2.6 remove()

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::remove (
    Iter const & i ) throw )    [inline]
```

Remove element pointed to by the iterator.

Definition at line 397 of file [list](#).

14.30.2.7 size()

```
template<typename D , template< typename A > class Alloc = New_allocator>
unsigned long cxx::List< D, Alloc >::size ( ) const throw )    [inline]
```

Get the length of the list.

Definition at line 401 of file [list](#).

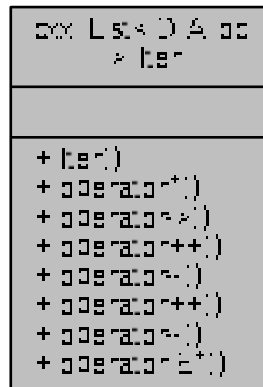
The documentation for this class was generated from the following file:

- [l4/cxx/list](#)

14.31 `cxx::List< D, Alloc >::Iter` Class Reference

Iterator.

Collaboration diagram for `cxx::List< D, Alloc >::Iter`:



Public Member Functions

- `operator E* () const throw ()`
operator for testing validity (syntactically equal to pointers)

14.31.1 Detailed Description

```
template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >::Iter
```

Iterator.

Forward and backward iterable.

Definition at line [354](#) of file [list](#).

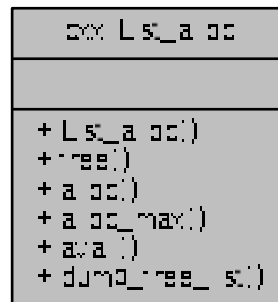
The documentation for this class was generated from the following file:

- `I4/cxx/list`

14.32 cxx::List_alloc Class Reference

Standard list-based allocator.

Collaboration diagram for cxx::List_alloc:



Public Member Functions

- [List_alloc](#) ()
Initializes an empty list allocator.
- void [free](#) (void *block, unsigned long size, bool initial_free=false)
Return a free memory block to the allocator.
- void * [alloc](#) (unsigned long size, unsigned align)
Alloc a memory block.
- void * [alloc_max](#) (unsigned long min, unsigned long *max, unsigned align, unsigned granularity)
Allocate a memory block of $min \leq size \leq max$.
- unsigned long [avail](#) ()
Get the amount of available memory.

14.32.1 Detailed Description

Standard list-based allocator.

Definition at line 31 of file [list_alloc](#).

14.32.2 Constructor & Destructor Documentation

14.32.2.1 List_alloc()

```
cxx::List_alloc::List_alloc ( ) [inline]
```

Initializes an empty list allocator.

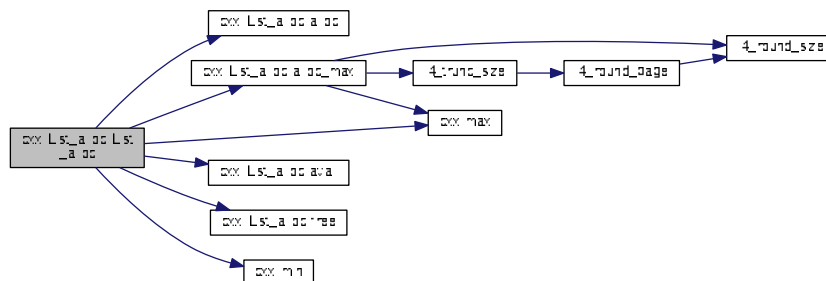
Note

To initialize the allocator with available memory use the [free\(\)](#) function.

Definition at line 56 of file [list_alloc](#).

References [alloc\(\)](#), [alloc_max\(\)](#), [avail\(\)](#), [L4::cerr](#), [free\(\)](#), [cxx::max\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:



14.32.3 Member Function Documentation

14.32.3.1 alloc()

```
void * cxx::List_alloc::alloc (
    unsigned long size,
    unsigned align ) [inline]
```

Alloc a memory block.

Parameters

<i>size</i>	Size of the memory block
<i>align</i>	Alignment constraint

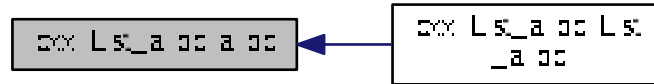
Returns

Pointer to memory block

Definition at line 354 of file [list_alloc](#).

Referenced by [List_alloc\(\)](#).

Here is the caller graph for this function:



14.32.3.2 `alloc_max()`

```
void * cxx::List_alloc::alloc_max (
    unsigned long min,
    unsigned long * max,
    unsigned align,
    unsigned granularity ) [inline]
```

Allocate a memory block of `min <= size <=max`.

Parameters

	<i>min</i>	Minimal size to allocate.
<i>in, out</i>	<i>max</i>	Maximum size to allocate. The actual allocated size is returned here.
	<i>align</i>	Alignment constraint.
	<i>granularity</i>	Granularity to use for the allocation.

Returns

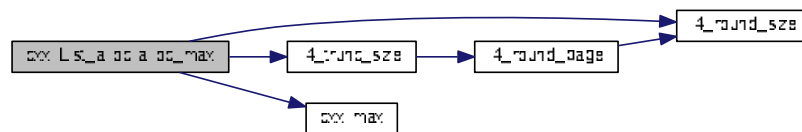
Pointer to memory block

Definition at line 257 of file [list_alloc](#).

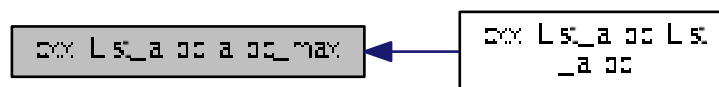
References [l4_round_size\(\)](#), [l4_trunc_size\(\)](#), and [cxx::max\(\)](#).

Referenced by [List_alloc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.32.3.3 avail()

```
unsigned long cxx::List_alloc::avail ( ) [inline]
```

Get the amount of available memory.

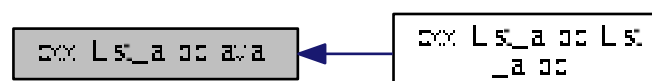
Returns

Available memory in bytes

Definition at line 425 of file [list_alloc](#).

Referenced by [List_alloc\(\)](#).

Here is the caller graph for this function:



14.32.3.4 free()

```
void cxx::List_alloc::free (
    void * block,
    unsigned long size,
    bool initial_free = false ) [inline]
```

Return a free memory block to the allocator.

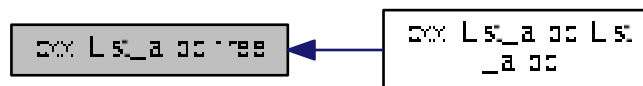
Parameters

<i>block</i>	pointer to memory block
<i>size</i>	size of memory block
<i>initial_free</i>	Set to true for putting fresh memory to the allocator. This will enforce alignment on that memory.

Definition at line 218 of file [list_alloc](#).

Referenced by [List_alloc\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

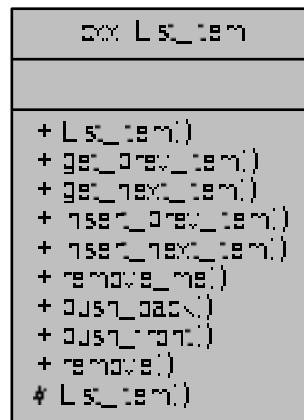
- `I4/cxx/list_alloc`

14.33 cxx::List_item Class Reference

Basic list item.

Inherited by `cxx::T_list_item< T >`.

Collaboration diagram for cxx::List_item:



Data Structures

- class [Iter](#)
Iterator for a list of ListItem-s.
- class [T_iter](#)
Iterator for derived classes from ListItem.

Public Member Functions

- [List_item](#) * [get_prev_item](#) () const throw ()
Get previous item.
- [List_item](#) * [get_next_item](#) () const throw ()
Get next item.
- void [insert_prev_item](#) ([List_item](#) *p) throw ()
Insert item p before this item.
- void [insert_next_item](#) ([List_item](#) *p) throw ()
Insert item p after this item.
- void [remove_me](#) () throw ()
Remove this item from the list.

Static Public Member Functions

- template<typename C , typename N >
static C * [push_back](#) (C *head, N *p) throw ()
Append item to a list.
- template<typename C , typename N >
static C * [push_front](#) (C *head, N *p) throw ()
Prepend item to a list.
- template<typename C , typename N >
static C * [remove](#) (C *head, N *p) throw ()
Remove item from a list.

14.33.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 37 of file [list](#).

14.33.2 Member Function Documentation

14.33.2.1 `get_next_item()`

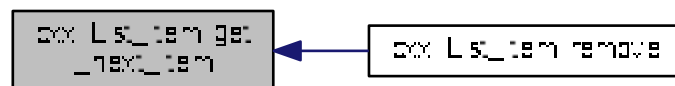
```
List_item* cxx::List_item::get_next_item ( ) const throw ( ) [inline]
```

Get next item.

Definition at line 176 of file [list](#).

Referenced by [remove\(\)](#).

Here is the caller graph for this function:



14.33.2.2 `get_prev_item()`

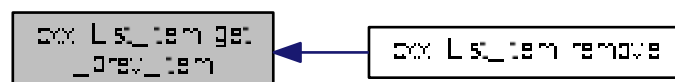
```
List_item* cxx::List_item::get_prev_item ( ) const throw ( ) [inline]
```

Get previous item.

Definition at line 173 of file [list](#).

Referenced by [remove\(\)](#).

Here is the caller graph for this function:



14.33.2.3 insert_next_item()

```
void cxx::List_item::insert_next_item (
    List_item * p ) throw ()    [inline]
```

Insert item *p* after this item.

Definition at line 189 of file [list](#).

14.33.2.4 insert_prev_item()

```
void cxx::List_item::insert_prev_item (
    List_item * p ) throw ()    [inline]
```

Insert item *p* before this item.

Definition at line 179 of file [list](#).

14.33.2.5 push_back()

```
template<typename C , typename N >
C * cxx::List_item::push_back (
    C * head,
    N * p ) throw ()    [inline], [static]
```

Append item to a list.

Convenience function for empty-head corner case.

Parameters

<i>head</i>	Pointer to the current list head.
<i>p</i>	Pointer to new item.

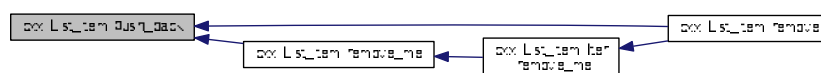
Returns

the pointer to the new head.

Definition at line 248 of file [list](#).

Referenced by [remove\(\)](#), and [remove_me\(\)](#).

Here is the caller graph for this function:



14.33.2.6 push_front()

```
template<typename C , typename N >
C * cxx::List_item::push_front (
    C * head,
    N * p ) throw ()    [inline], [static]
```

Prepend item to a list.

Convenience function for empty-head corner case.

Parameters

<i>head</i>	pointer to the current list head.
<i>p</i>	pointer to new item.

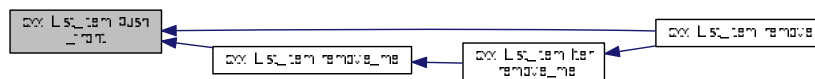
Returns

the pointer to the new head.

Definition at line 259 of file [list](#).

Referenced by [remove\(\)](#), and [remove_me\(\)](#).

Here is the caller graph for this function:



14.33.2.7 remove()

```
template<typename C , typename N >
C * cxx::List_item::remove (
    C * head,
    N * p ) throw ()    [inline], [static]
```

Remove item from a list.

Convenience function for remove-head corner case.

Parameters

<i>head</i>	pointer to the current list head.
<i>p</i>	pointer to the item to remove.

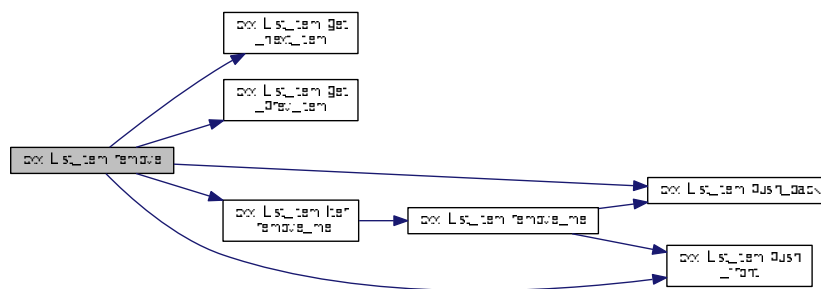
Returns

the pointer to the new head.

Definition at line 269 of file [list](#).

References [get_next_item\(\)](#), [get_prev_item\(\)](#), [push_back\(\)](#), [push_front\(\)](#), and [cxx::List_item::Iter::remove_me\(\)](#).

Here is the call graph for this function:



14.33.2.8 remove_me()

```
void cxx::List_item::remove_me ( ) throw ( ) [inline]
```

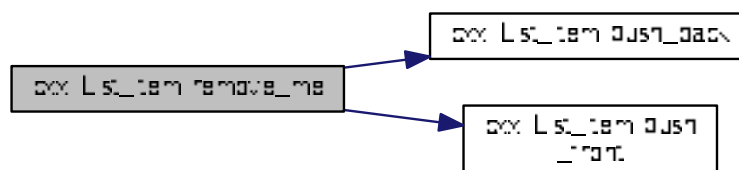
Remove this item from the list.

Definition at line 198 of file [list](#).

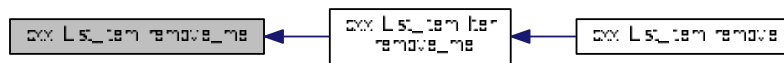
References [push_back\(\)](#), and [push_front\(\)](#).

Referenced by [cxx::List_item::Iter::remove_me\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



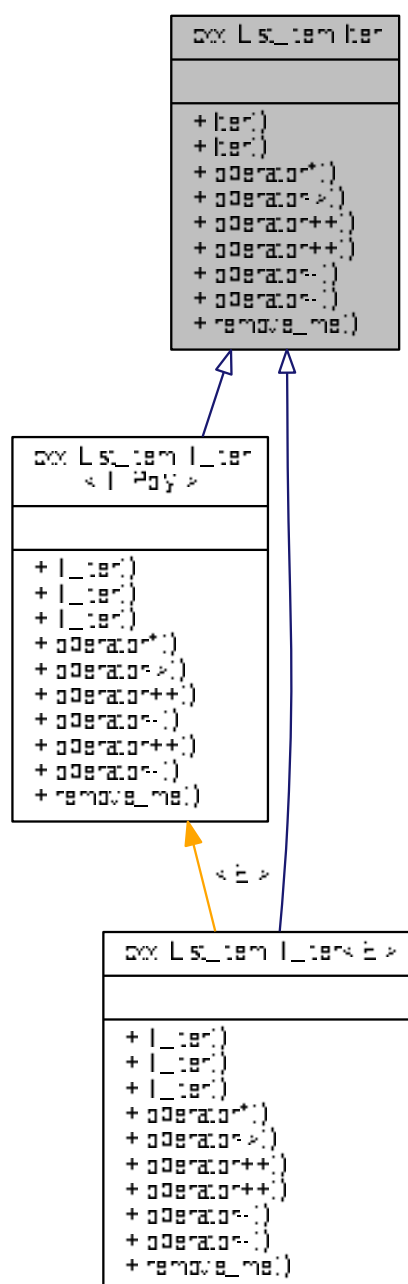
The documentation for this class was generated from the following file:

- `I4/cxx/list`

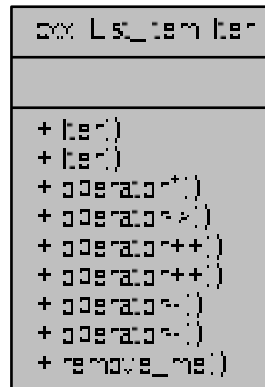
14.34 `cxx::List_item::Iter` Class Reference

Iterator for a list of `ListItem`-s.

Inheritance diagram for cxx::List_item::Iter:



Collaboration diagram for `cxx::List_item::Iter`:



Public Member Functions

- [List_item * remove_me \(\) throw \(\)](#)
Remove item pointed to by iterator, and return pointer to element.

14.34.1 Detailed Description

Iterator for a list of `ListItem`-s.

The Iterator iterates till it finds the first element again.

Definition at line [45](#) of file [list](#).

14.34.2 Member Function Documentation

14.34.2.1 `remove_me()`

```
List_item* cxx::List_item::Iter::remove_me ( ) throw () [inline]
```

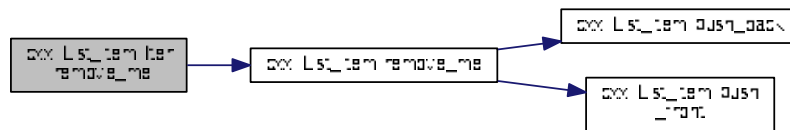
Remove item pointed to by iterator, and return pointer to element.

Definition at line [86](#) of file [list](#).

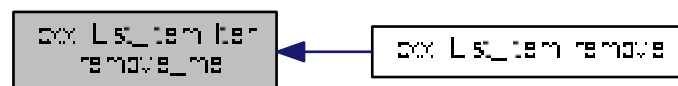
References [cxx::List_item::remove_me\(\)](#).

Referenced by [cxx::List_item::remove\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



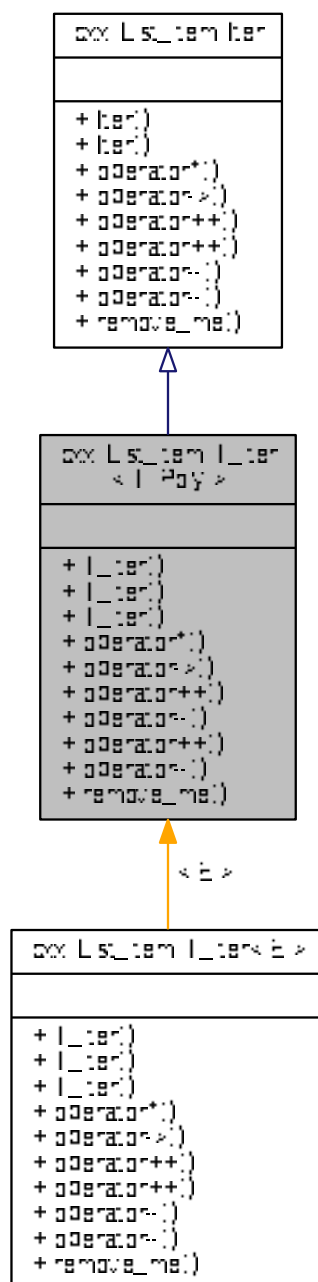
The documentation for this class was generated from the following file:

- `I4/cxx/list`

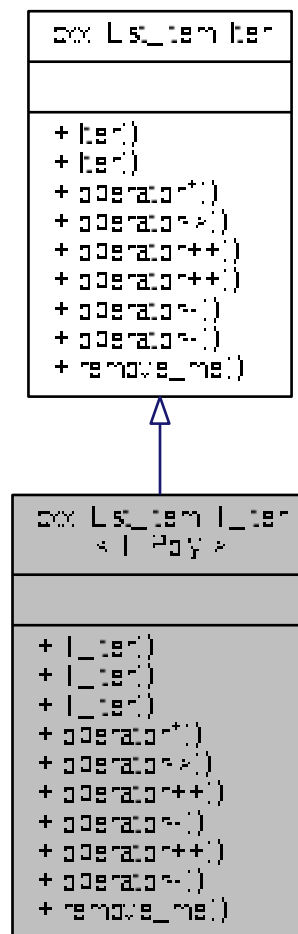
14.35 cxx::List_item::T_iter< T, Poly > Class Template Reference

Iterator for derived classes from `ListItem`.

Inheritance diagram for `cxx::List_item::T_iter< T, Poly >`:



Collaboration diagram for cxx::List_item::T_iter< T, Poly >:



Additional Inherited Members

14.35.1 Detailed Description

```
template<typename T, bool Poly = false>
class cxx::List_item::T_iter< T, Poly >
```

Iterator for derived classes from ListItem.

Allows direct access to derived classes by * operator.

Example: `class Foo : public ListItem { public: typedef T_iter<Foo> lter; ... };`

Definition at line 119 of file [list](#).

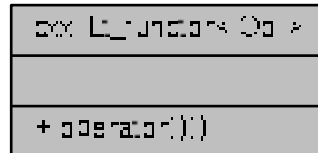
The documentation for this class was generated from the following file:

- `I4/cxx/list`

14.36 `cxx::Lt_functor< Obj >` Struct Template Reference

Generic comparator class that defaults to the less-than operator.

Collaboration diagram for `cxx::Lt_functor< Obj >`:



14.36.1 Detailed Description

```
template<typename Obj>
struct cxx::Lt_functor< Obj >
```

Generic comparator class that defaults to the less-than operator.

Definition at line 29 of file [std_ops](#).

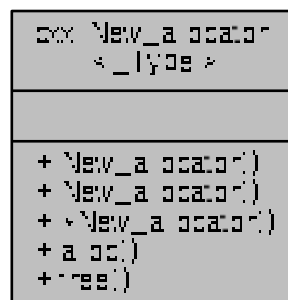
The documentation for this struct was generated from the following file:

- `I4/cxx/std_ops`

14.37 `cxx::New_allocator< _Type >` Class Template Reference

Standard allocator based on `operator new ()`.

Collaboration diagram for `cxx::New_allocator< _Type >`:



14.37.1 Detailed Description

```
template<typename _Type>
class cxx::New_allocator<_Type >
```

Standard allocator based on `operator new ()` .

This allocator is the default allocator used for the *cxx Containers*, such as `cxx::Avl_set` and `cxx::Avl_map`, to allocate the internal data structures.

Definition at line 60 of file `std_alloc`.

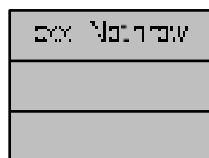
The documentation for this class was generated from the following file:

- `I4/cxx/std_alloc`

14.38 `cxx::Nothrow` Class Reference

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Collaboration diagram for `cxx::Nothrow`:



14.38.1 Detailed Description

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Definition at line 30 of file `std_alloc`.

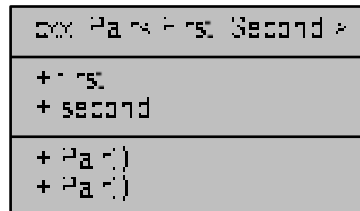
The documentation for this class was generated from the following file:

- `I4/cxx/std_alloc`

14.39 cxx::Pair< First, Second > Struct Template Reference

Pair of two values.

Collaboration diagram for cxx::Pair< First, Second >:



Public Types

- typedef First [First_type](#)
Type of first value.
- typedef Second [Second_type](#)
Type of second value.

Public Member Functions

- template<typename A1 , typename A2 >
[Pair](#) (A1 &&[first](#), A2 &&[second](#))
Create a pair from the two values.
- [Pair](#) ()
Default construction.

Data Fields

- First [first](#)
First value.
- Second [second](#)
Second value.

14.39.1 Detailed Description

```
template<typename First, typename Second>
struct cxx::Pair< First, Second >
```

Pair of two values.

Standard container for a pair of values.

Parameters

<i>First</i>	Type of the first value.
<i>Second</i>	Type of the second value.

Definition at line 36 of file [pair](#).

14.39.2 Constructor & Destructor Documentation

14.39.2.1 `Pair()`

```
template<typename First, typename Second>
template<typename A1 , typename A2 >
cxx::Pair< First, Second >::Pair (
    A1 && first,
    A2 && second ) [inline]
```

Create a pair from the two values.

Parameters

<i>first</i>	The first value.
<i>second</i>	The second value.

Definition at line 54 of file [pair](#).

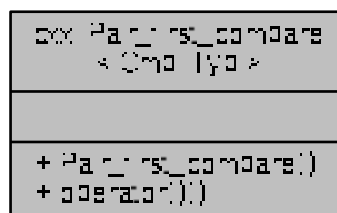
The documentation for this struct was generated from the following file:

- [l4/cxx/pair](#)

14.40 `cxx::Pair_first_compare< Cmp, Typ >` Class Template Reference

Comparison functor for [Pair](#).

Collaboration diagram for `cxx::Pair_first_compare< Cmp, Typ >`:



Public Member Functions

- [Pair_first_compare](#) (Cmp const &cmp=Cmp())
Construction.
- bool [operator\(\)](#) (Typ const &l, Typ const &r) const
Do the comaprison based on the first value.

14.40.1 Detailed Description

```
template<typename Cmp, typename Typ>
class cxx::Pair_first_compare< Cmp, Typ >
```

Comparison functor for [Pair](#).

Parameters

<i>Cmp</i>	Comparison functor for the first value of the pair.
<i>Typ</i>	The pair type.

This functor can be used to compare [Pair](#) values with respect to the first value.

Definition at line 75 of file [pair](#).

14.40.2 Constructor & Destructor Documentation

14.40.2.1 Pair_first_compare()

```
template<typename Cmp , typename Typ >
cxx::Pair_first_compare< Cmp, Typ >::Pair_first_compare (
    Cmp const & cmp = Cmp() ) [inline]
```

Construction.

Parameters

<i>cmp</i>	The comparison functor used for the first value.
------------	--

Definition at line 85 of file [pair](#).

14.40.3 Member Function Documentation

14.40.3.1 operator()

```
template<typename Cmp , typename Typ >
bool cxx::Pair_first_compare< Cmp, Typ >::operator() (
    Typ const & l,
    Typ const & r ) const [inline]
```

Do the comparison based on the first value.

Parameters

<i>l</i>	The lefthand value.
<i>r</i>	The righthand value.

Definition at line 92 of file [pair](#).

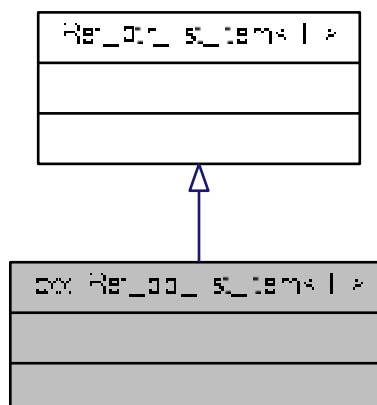
The documentation for this class was generated from the following file:

- [l4/cxx/pair](#)

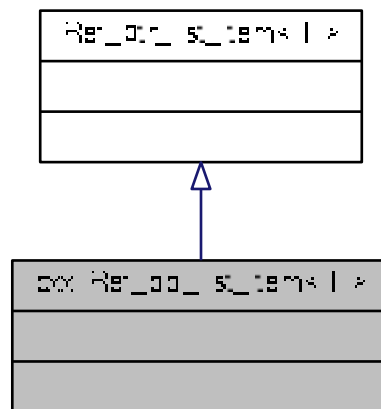
14.41 cxx::Ref_obj_list_item< T > Struct Template Reference

Item for list linked via [cxx::Ref_ptr](#) with default reference counting.

Inheritance diagram for cxx::Ref_obj_list_item< T >:



Collaboration diagram for `cxx::Ref_obj_list_item< T >`:



14.41.1 Detailed Description

```
template<typename T>
struct cxx::Ref_obj_list_item< T >
```

Item for list linked via [cxx::Ref_ptr](#) with default reference counting.

Definition at line 26 of file [ref_ptr_list](#).

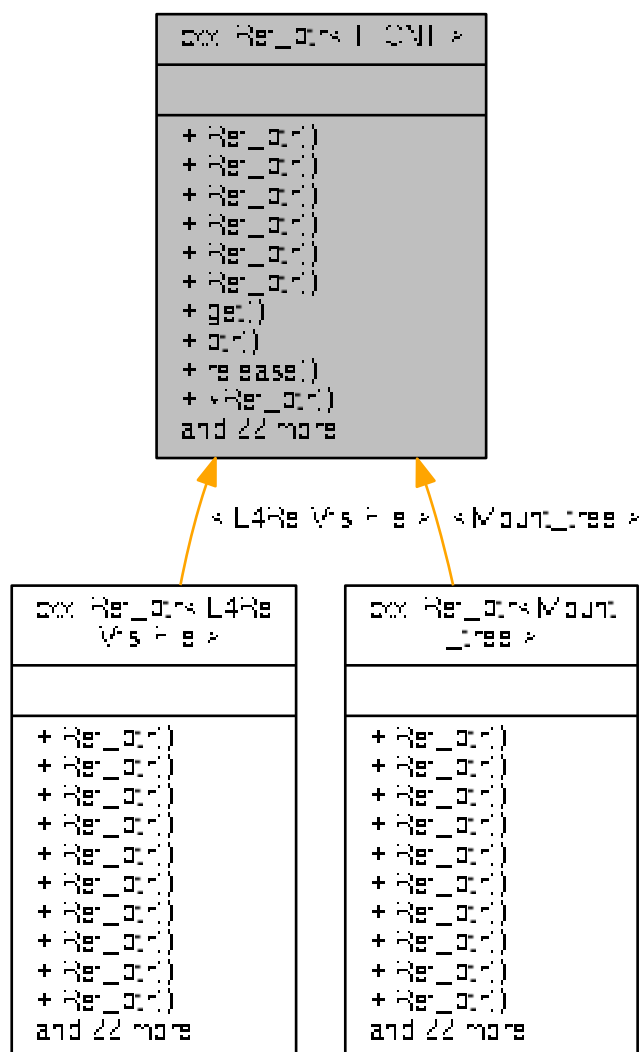
The documentation for this struct was generated from the following file:

- `I4/cxx/ref_ptr_list`

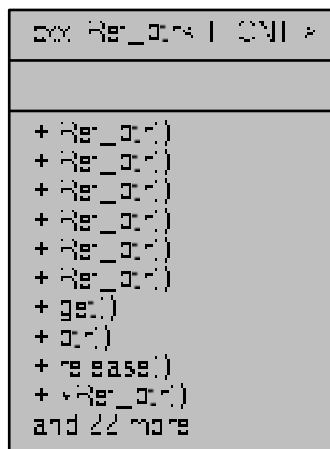
14.42 `cxx::Ref_ptr< T, CNT >` Class Template Reference

A reference-counting pointer with automatic cleanup.

Inheritance diagram for cxx::Ref_ptr< T, CNT >:



Collaboration diagram for `cxx::Ref_ptr< T, CNT >`:



Public Member Functions

- [Ref_ptr](#) () throw ()
Default constructor creates a pointer with no managed object.
- [Ref_ptr](#) (Wp const &o) throw ()
Create a shared pointer from a weak pointer.
- [Ref_ptr](#) (decltype(nullptr) n) noexcept
allow creation from `nullptr`
- `template<typename X >`
[Ref_ptr](#) (X *o) throw ()
Create a shared pointer from a raw pointer.
- [Ref_ptr](#) (T *o, bool d) throw ()
Create a shared pointer from a raw pointer without creating a new reference.
- T * [get](#) () const throw ()
Return a raw pointer to the object this shared pointer points to.
- T * [ptr](#) () const throw ()
Return a raw pointer to the object this shared pointer points to.
- T * [release](#) () throw ()
Release the shared pointer without removing the reference.

14.42.1 Detailed Description

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
class cxx::Ref_ptr< T, CNT >
```

A reference-counting pointer with automatic cleanup.

Template Parameters

<i>T</i>	Type of object the pointer points to.
<i>CNT</i>	Type of management class that manages the life time of the object.

This pointer is similar to the standard C++-11 `shared_ptr` but it does the reference counting directly in the object being pointed to, so that no additional management structures need to be allocated from the heap.

Classes that use this pointer type must implement two functions:

```
int remove_ref()
```

is called when a reference is removed and must return 0 when there are no further references to the object.

```
void add_ref()
```

is called when another `ref_ptr` to the object is created.

`Ref_obj` provides a simple implementation of this interface from which classes may inherit.

Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 80 of file [ref_ptr](#).

14.42.2 Constructor & Destructor Documentation

14.42.2.1 `Ref_ptr()` [1/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    Wp const & o ) throw )    [inline]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 101 of file [ref_ptr](#).

14.42.2.2 Ref_ptr() [2/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
template<typename X >
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    X * o ) throw )    [inline], [explicit]
```

Create a shared pointer from a raw pointer.

In contrast to C++11 `shared_ptr` it is safe to use this constructor multiple times and have the same reference counter.

Definition at line 115 of file `ref_ptr`.

14.42.2.3 Ref_ptr() [3/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    T * o,
    bool d ) throw )    [inline]
```

Create a shared pointer from a raw pointer without creating a new reference.

Parameters

<i>o</i>	Pointer to the object.
<i>d</i>	Dummy parameter to select this constructor at compile time. The value may be true or false.

This is the counterpart to `release()`.

Definition at line 138 of file `ref_ptr`.

14.42.3 Member Function Documentation**14.42.3.1 get()**

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::get ( ) const throw )    [inline]
```

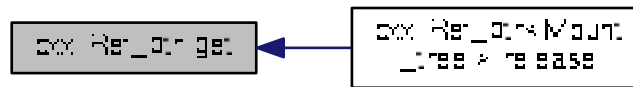
Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 145 of file `ref_ptr`.

Referenced by `cxx::Ref_ptr< Mount_tree >::release()`.

Here is the caller graph for this function:



14.42.3.2 `ptr()`

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::ptr ( ) const throw ( ) [inline]
```

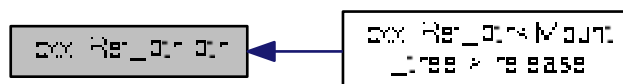
Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 151 of file `ref_ptr`.

Referenced by `cxx::Ref_ptr< Mount_tree >::release()`.

Here is the caller graph for this function:



14.42.3.3 `release()`

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::release ( ) throw ( ) [inline]
```

Release the shared pointer without removing the reference.

Returns

A raw pointer to the managed object.

Definition at line 162 of file [ref_ptr](#).

Referenced by [cxx::Ref_ptr< Mount_tree >::release\(\)](#).

Here is the caller graph for this function:



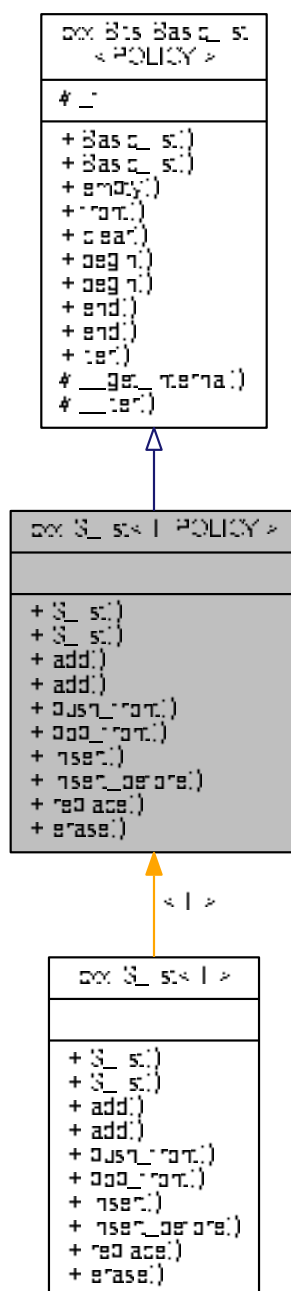
The documentation for this class was generated from the following file:

- [l4/cxx/ref_ptr](#)

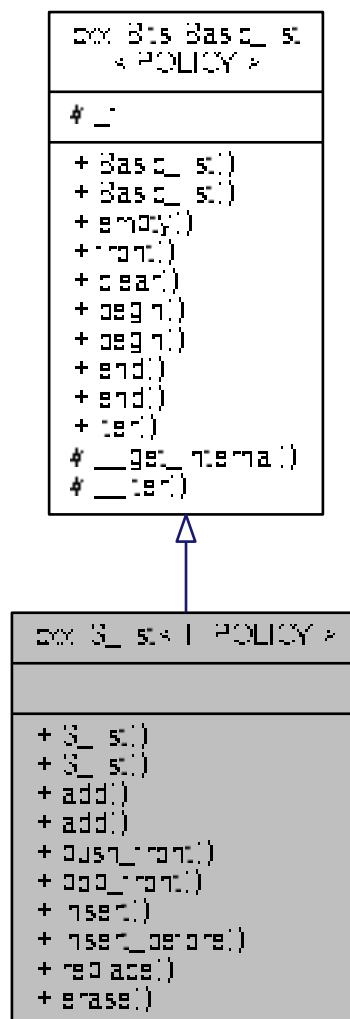
14.43 `cxx::S_list< T, POLICY >` Class Template Reference

Simple single-linked list.

Inheritance diagram for cxx::S_list< T, POLICY >:



Collaboration diagram for `cxx::S_list< T, POLICY >`:



Public Member Functions

- void `add` (T *e)
Add an element to the front of the list.
- void `push_front` (T *e)
Add an element to the front of the list.
- T * `pop_front` ()
Remove and return the head element of the list.

Additional Inherited Members

14.43.1 Detailed Description

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
class cxx::S_list< T, POLICY >
```

Simple single-linked list.

Template Parameters

<code>T</code>	Type of elements saved in the list. Must inherit from <code>cxx::S_list_item</code>
----------------	---

Definition at line 50 of file [slist](#).

14.43.2 Member Function Documentation

14.43.2.1 `pop_front()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
T* cxx::S_list< T, POLICY >::pop_front ( ) [inline]
```

Remove and return the head element of the list.

Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 91 of file [slist](#).

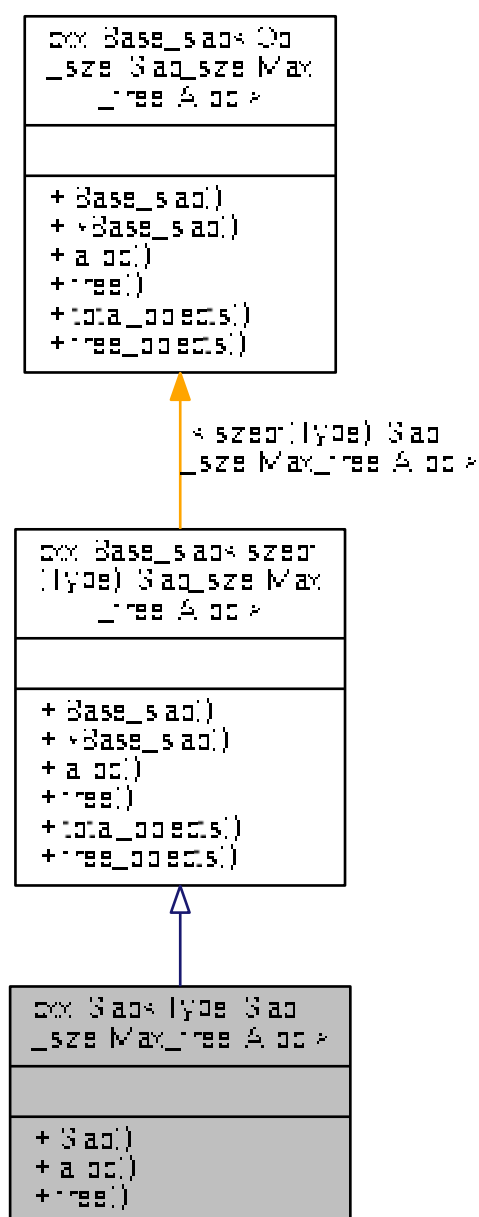
The documentation for this class was generated from the following file:

- `I4/cxx/slist`

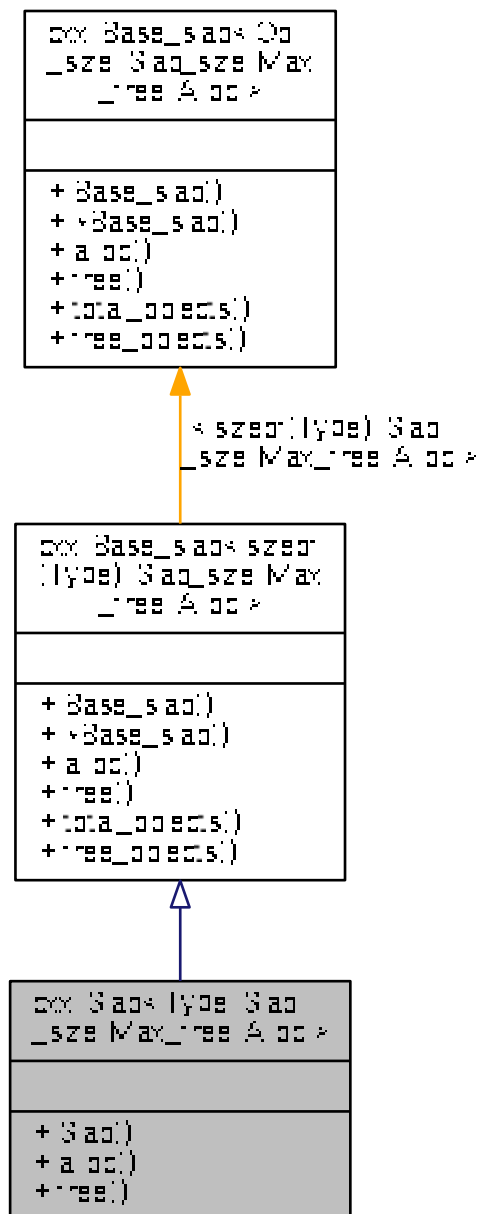
14.44 `cxx::Slab< Type, Slab_size, Max_free, Alloc >` Class Template Reference

[Slab](#) allocator for object of type *Type*.

Inheritance diagram for `cxx::Slab< Type, Slab_size, Max_free, Alloc >`:



Collaboration diagram for `cxx::Slab< Type, Slab_size, Max_free, Alloc >`:



Public Member Functions

- `Type * alloc ()` throw ()
Allocate an object of type *Type*.
- `void free (Type *o)` throw ()
Free the object addressed by *o*.

Additional Inherited Members

14.44.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>
class cxx::Slab< Type, Slab_size, Max_free, Alloc >
```

[Slab](#) allocator for object of type *Type*.

Parameters

<i>Type</i>	the type of the objects to manage.
<i>Slab_size</i>	size of a slab cache.
<i>Max_free</i>	the maximum number of free slab caches.
<i>Alloc</i>	the allocator for the slab caches.

Definition at line 297 of file [slab_alloc](#).

14.44.2 Member Function Documentation

14.44.2.1 alloc()

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
Type* cxx::Slab< Type, Slab_size, Max_free, Alloc >::alloc ( ) throw )    [inline]
```

Allocate an object of type *Type*.

Returns

A pointer to the object just allocated, or 0 on failure.

Definition at line 314 of file [slab_alloc](#).

14.44.2.2 free()

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
void cxx::Slab< Type, Slab_size, Max_free, Alloc >::free (
    Type * o ) throw )    [inline]
```

Free the object addressed by *o*.

Parameters

<code>o</code>	The pointer to the object to free.
----------------	------------------------------------

Precondition

The object must have been allocated with this allocator.

Definition at line 325 of file [slab_alloc](#).

References [L4_PAGESIZE](#).

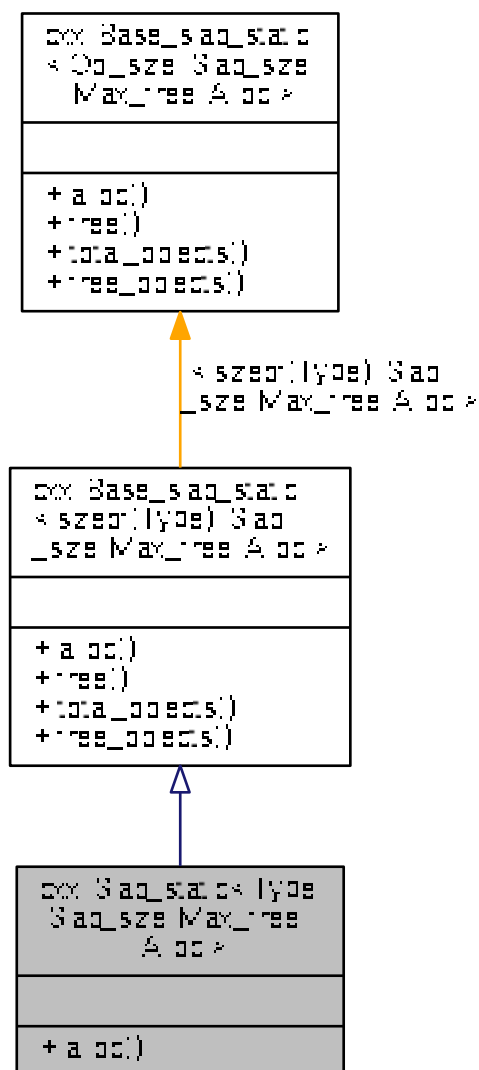
The documentation for this class was generated from the following file:

- `l4/cxx/slab_alloc`

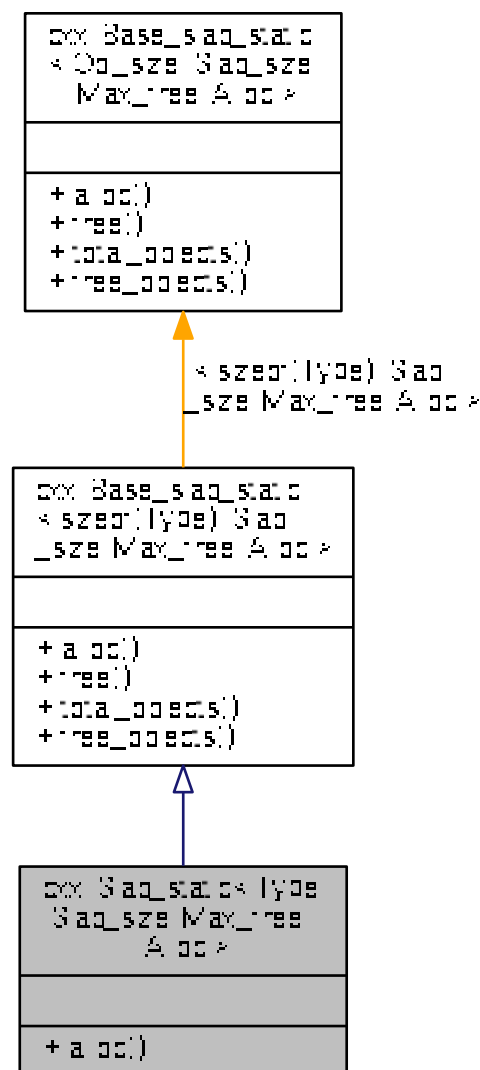
14.45 `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >` Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

Inheritance diagram for `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >`:



Collaboration diagram for `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >`:



Public Member Functions

- `Type * alloc () throw ()`
Allocate an object of type *Type*.

14.45.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>
class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

Parameters

<i>Type</i>	The type of the objects to manage.
<i>Slab_size</i>	The size of a slab cache.
<i>Max_free</i>	The maximum number of free slab caches.
<i>Alloc</i>	The allocator for the slab caches.

This slab allocator class is useful for merging slab allocators with the same parameters (equal *sizeof(Type)*, *Slab_size*, *Max_free*, and *Alloc* parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 415 of file [slab_alloc](#).

14.45.2 Member Function Documentation**14.45.2.1 alloc()**

```
template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A
> class Alloc = New_allocator>
Type* cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc ( ) throw ( )    [inline]
```

Allocate an object of type *Type*.

Returns

A pointer to the just allocated object, or 0 of failure.

Definition at line 425 of file [slab_alloc](#).

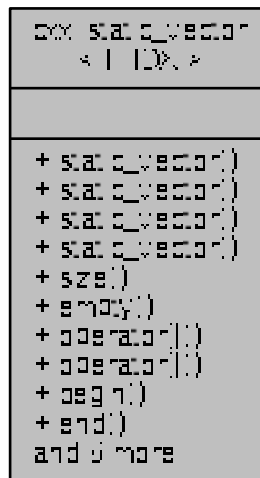
The documentation for this class was generated from the following file:

- I4/cxx/slab_alloc

14.46 cxx::static_vector< T, IDX > Class Template Reference

Simple encapsulation for a dynamically allocated array.

Collaboration diagram for `cxx::static_vector< T, IDX >`:



Public Member Functions

- `template<typename X, typename = typename enable_if<is_convertible<X, T>::value>::type>`
`static_vector (static_vector< X, IDX > const &o)`
Conversion from compatible arrays.
- `index_type index (value_type const *o) const`
Get the index of the given element of the array.

14.46.1 Detailed Description

```

template<typename T, typename IDX = unsigned>
class cxx::static_vector< T, IDX >
  
```

Simple encapsulation for a dynamically allocated array.

The main purpose of this class is to support C++11 range for for simple dynamically allocated array with static size.

Definition at line 16 of file [static_vector](#).

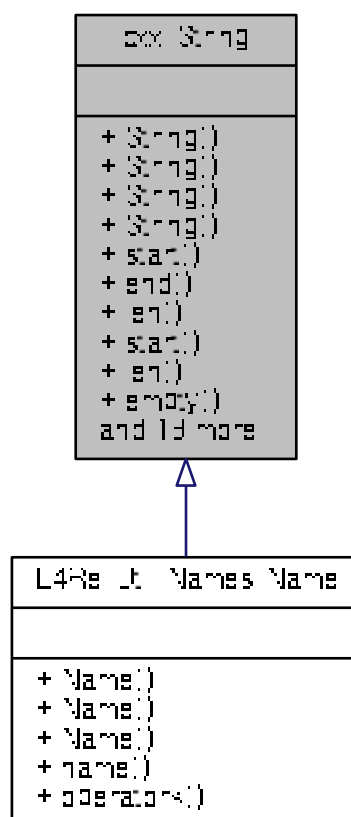
The documentation for this class was generated from the following file:

- `I4/cxx/static_vector`

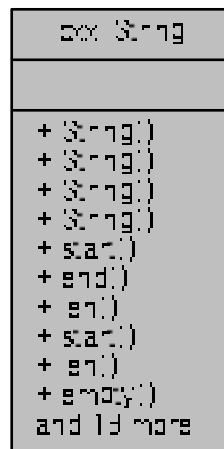
14.47 cxx::String Class Reference

Allocation free string class with explicit length field.

Inheritance diagram for cxx::String:



Collaboration diagram for cxx::String:



Public Types

- typedef char const * [Index](#)
Character index type.

Public Member Functions

- [String](#) (char const *s) throw ()
Initialize from a zero-terminated string.
- [String](#) (char const *s, unsigned long len) throw ()
Initialize from a pointer to first character and a length.
- [String](#) (char const *s, char const *e) throw ()
Initialize with start and end pointer.
- [String](#) ()
Zero-initialize. Create an invalid string.
- [Index start](#) () const
Pointer to first character.
- [Index end](#) () const
Pointer to first byte behind the string.
- int [len](#) () const
Length.
- void [start](#) (char const *s)
Set start.
- void [len](#) (unsigned long len)
Set length.
- bool [empty](#) () const
Check if the string has length zero.
- [String head](#) ([Index end](#)) const

- Return prefix up to index.*
- **String head** (unsigned long **end**) const
 - Prefix of length **end**.*
- **String substr** (unsigned long **idx**, unsigned long **len**=~0UL) const
 - Substring of length **len** starting at **idx**.*
- **String substr** (char const ***start**, unsigned long **len**=0) const
 - Substring of length **len** starting at **start**.*
- template<typename F >
 - char const * **find_match** (F &&match) const
 - Find matching character. **match** should be a function such as `isspace`.*
- char const * **find** (char const ***c**) const
 - Find character. Return **end()** if not found.*
- char const * **find** (int **c**) const
 - Find character. Return **end()** if not found.*
- char const * **rfind** (char const ***c**) const
 - Find right-most character. Return **end()** if not found.*
- **Index starts_with** (cxx::String const &**c**) const
 - Check if **c** is a prefix of string.*
- char const * **find** (int **c**, char const ***s**) const
 - Find character **c** starting at position **s**. Return **end()** if not found.*
- char const * **find** (char const ***c**, char const ***s**) const
 - Find character set at position.*
- char const & **operator[]** (unsigned long **idx**) const
 - Get character at **idx**.*
- char const & **operator[]** (int **idx**) const
 - Get character at **idx**.*
- char const & **operator[]** (**Index** **idx**) const
 - Get character at **idx**.*
- bool **eof** (char const ***s**) const
 - Check if pointer **s** points behind string.*
- template<typename INT >
 - int **from_dec** (INT ***v**) const
 - Convert decimal string to integer.*
- template<typename INT >
 - int **from_hex** (INT ***v**) const
 - Convert hex string to integer.*
- bool **operator==** (**String** const &**o**) const
 - Equality.*
- bool **operator!=** (**String** const &**o**) const
 - Inequality.*

14.47.1 Detailed Description

Allocation free string class with explicit length field.

This class is used to group characters of a string which belong to one syntactical token types number, identifier, string, whitespace or another single character.

Strings in this class can contain null bytes and may denote parts of other strings.

Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 41 of file [string](#).

14.47.2 Constructor & Destructor Documentation

14.47.2.1 String()

```
cxx::String::String (
    char const * s,
    char const * e ) throw ()    [inline]
```

Initialize with start and end pointer.

Parameters

<i>s</i>	first character of the string
<i>e</i>	pointer to first byte behind the string

Definition at line 59 of file [string](#).

14.47.3 Member Function Documentation

14.47.3.1 find()

```
char const* cxx::String::find (
    char const * c,
    char const * s ) const    [inline]
```

Find character set at position.

Parameters

<i>c</i>	zero-terminated string of characters to search for
<i>s</i>	start position of search in string

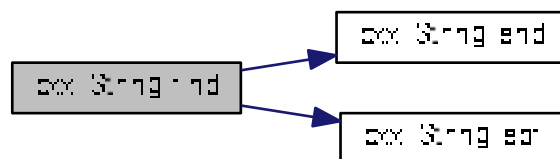
Return values

end()	if no char in <i>c</i> is contained in string at or behind <i>s</i> .
<i>position</i>	in string of some character in <i>c</i> .

Definition at line 202 of file [string](#).

References [end\(\)](#), and [eof\(\)](#).

Here is the call graph for this function:



14.47.3.2 from_dec()

```
template<typename INT >
int cxx::String::from_dec (
    INT * v ) const [inline]
```

Convert decimal string to integer.

Template Parameters

<i>INT</i>	result integer type
------------	---------------------

Parameters

out	<i>v</i>	conversion result
-----	----------	-------------------

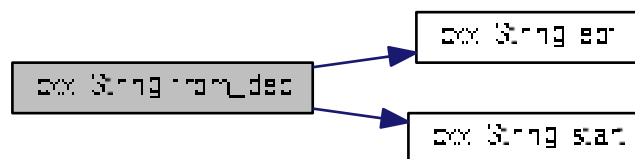
Returns

position of first character not converted.

Definition at line [239](#) of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



14.47.3.3 from_hex()

```
template<typename INT >  
int cxx::String::from_hex (  
    INT * v ) const [inline]
```

Convert hex string to integer.

Template Parameters

<i>INT</i>	result integer type
------------	---------------------

Parameters

out	<i>v</i>	conversion result
-----	----------	-------------------

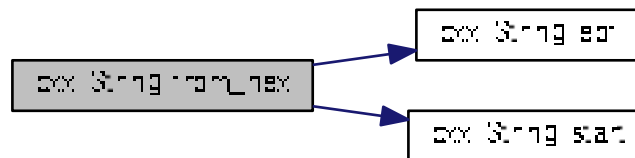
Return values

<i>-1</i>	if the maximal amount of digits fitting into <i>INT</i> have been read,
<i>position</i>	of first character not converted otherwise.

Definition at line 268 of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



14.47.3.4 starts_with()

Index `cxx::String::starts_with (`
 `cxx::String const & c) const [inline]`

Check if `c` is a prefix of string.

Returns

0 if `c` is not a prefix, if it is a prefix, return first position not in `c` (which might be `end()`).

Definition at line 166 of file [string](#).

References [start\(\)](#).

Here is the call graph for this function:



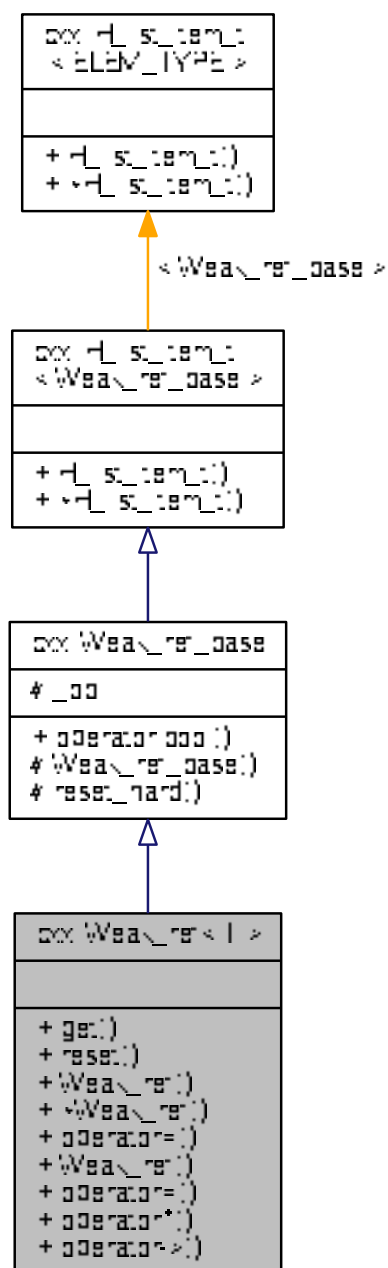
The documentation for this class was generated from the following file:

- `l4/cxx/string`

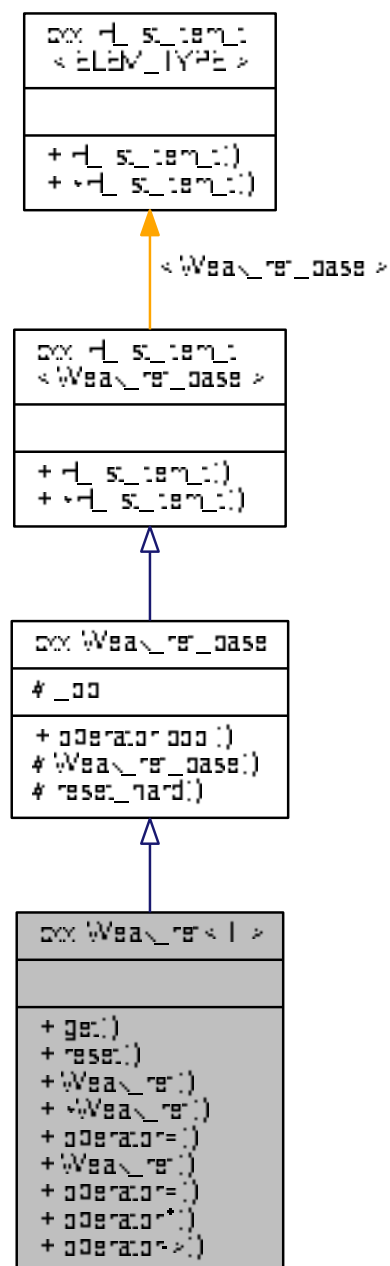
14.48 cxx::Weak_ref< T > Class Template Reference

Typed weak reference to an object of type T.

Inheritance diagram for cxx::Weak_ref< T >:



Collaboration diagram for `cxx::Weak_ref< T >`:



Additional Inherited Members

14.48.1 Detailed Description

```
template<typename T>
class cxx::Weak_ref< T >
```

Typed weak reference to an object of type `T`.

Template Parameters

<i>T</i>	The type of the referenced object.
----------	------------------------------------

A weak reference is a reference that is invalidated when the referenced object is about to be deleted. All weak references to an object are kept in a linked list and all the weak references are iterated and reset by the `Weak_ref_base::List` destructor or `Weak_ref_base::reset()`.

The type `T` must provide two methods that handle the housekeeping of weak references: `remove_weak_ref(Weak_ref_base *)` and `add_weak_ref(Weak_ref_base *)`. These functions must handle the insertion and removal of the weak reference into the respective `Weak_ref_base::List` object. For convenience one can use the `cxx::Weak_ref_obj` as a base class that handles weak references for you.

Definition at line 67 of file [weak_ref](#).

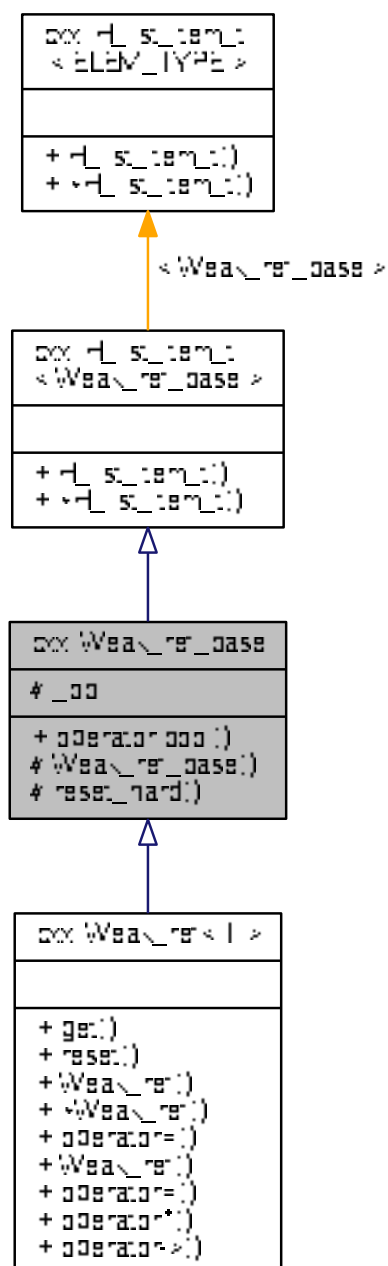
The documentation for this class was generated from the following file:

- `I4/cxx/weak_ref`

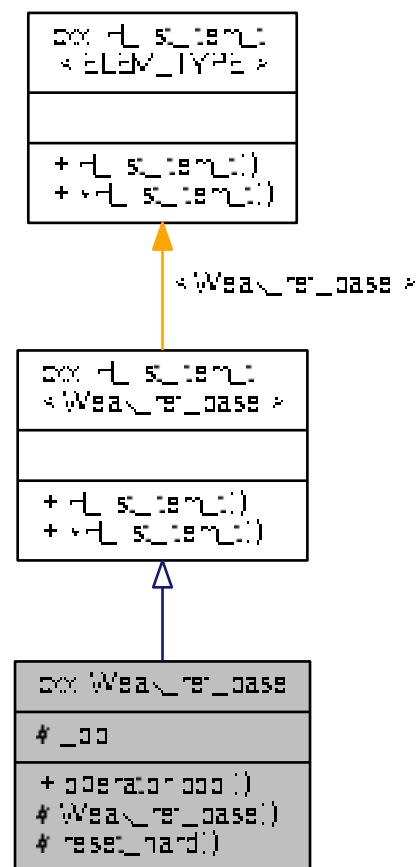
14.49 cxx::Weak_ref_base Class Reference

Generic (base) weak reference to some object.

Inheritance diagram for `cxx::Weak_ref_base`:



Collaboration diagram for cxx::Weak_ref_base:



Additional Inherited Members

14.49.1 Detailed Description

Generic (base) weak reference to some object.

A weak reference is a reference that gets reset to NULL when the object shall be deleted. All weak references to the same object are kept in a linked list of weak references.

For typed weak references see [cxx::Weak_ref](#).

Definition at line 25 of file [weak_ref](#).

The documentation for this class was generated from the following file:

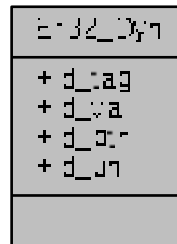
- `I4/cxx/weak_ref`

14.50 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Dyn:



Data Fields

- [Elf32_Word d_tag](#)
see DT_ values
- [Elf32_Word d_val](#)
integer values with various interpret.
- [Elf32_Addr d_ptr](#)
program virtual addresses

14.50.1 Detailed Description

ELF32 dynamic entry.

Definition at line 500 of file [elf.h](#).

14.50.2 Field Documentation

14.50.2.1 d_val

[Elf32_Word](#) Elf32_Dyn::d_val

integer values with various interpret.

Definition at line 503 of file [elf.h](#).

The documentation for this struct was generated from the following file:

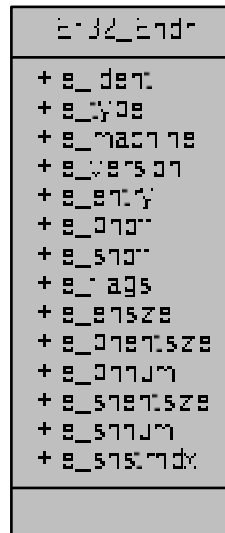
- [l4/util/elf.h](#)

14.51 Elf32_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Ehdr:



Data Fields

- [Elf32_Half e_type](#)
type of ELF file
- [Elf32_Half e_machine](#)
required architecture
- [Elf32_Word e_version](#)
file version
- [Elf32_Addr e_entry](#)
initial eip
- [Elf32_Off e_phoff](#)
offset of program header table
- [Elf32_Off e_shoff](#)
offset of file header table
- [Elf32_Word e_flags](#)
processor-specific flags
- [Elf32_Half e_ehsize](#)
size of ELF header
- [Elf32_Half e_phentsize](#)
size of program header entry
- [Elf32_Half e_phnum](#)

of entries in prog.

- [Elf32_Half e_shentsize](#)
size of section header entry
- [Elf32_Half e_shnum](#)

of entries in sect.

- [Elf32_Half e_shstrndx](#)
sect.head.tab.idx of strtab

14.51.1 Detailed Description

ELF32 header.

Definition at line [122](#) of file [elf.h](#).

14.51.2 Field Documentation

14.51.2.1 e_phnum

[Elf32_Half](#) [Elf32_Ehdr::e_phnum](#)

of entries in prog.

head. tab.

Definition at line [133](#) of file [elf.h](#).

14.51.2.2 e_shnum

[Elf32_Half](#) [Elf32_Ehdr::e_shnum](#)

of entries in sect.

head. tab.

Definition at line [135](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

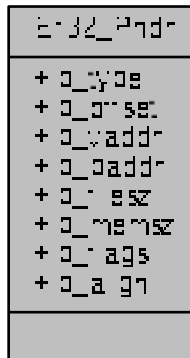
- [l4/util/elf.h](#)

14.52 Elf32_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Phdr:



Data Fields

- [Elf32_Word p_type](#)
type of program section
- [Elf32_Off p_offset](#)
file offset of program section
- [Elf32_Addr p_vaddr](#)
memory address of prog section
- [Elf32_Addr p_paddr](#)
physical address (ignored)
- [Elf32_Word p_filesz](#)
file size of program section
- [Elf32_Word p_memsz](#)
memory size of program section
- [Elf32_Word p_flags](#)
flags
- [Elf32_Word p_align](#)
alignment of section

14.52.1 Detailed Description

ELF32 program header.

Definition at line 419 of file [elf.h](#).

The documentation for this struct was generated from the following file:

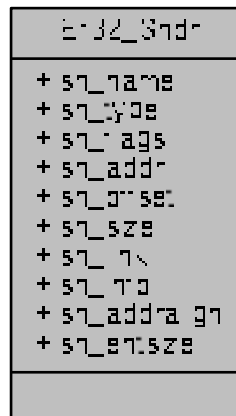
- [l4/util/elf.h](#)

14.53 Elf32_Shdr Struct Reference

ELF32 section header - figure 1-9, page 1-9.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Shdr:



Data Fields

- [Elf32_Word sh_name](#)
name of sect (idx into strtab)
- [Elf32_Word sh_type](#)
section's type
- [Elf32_Word sh_flags](#)
section's flags
- [Elf32_Addr sh_addr](#)
memory address of section
- [Elf32_Off sh_offset](#)
file offset of section
- [Elf32_Word sh_size](#)
file size of section
- [Elf32_Word sh_link](#)
idx to associated header section
- [Elf32_Word sh_info](#)
extra info of header section
- [Elf32_Word sh_addralign](#)
address alignment constraints
- [Elf32_Word sh_entsize](#)
size of entry if sect is table

14.53.1 Detailed Description

ELF32 section header - figure 1-9, page 1-9.

Definition at line 342 of file [elf.h](#).

The documentation for this struct was generated from the following file:

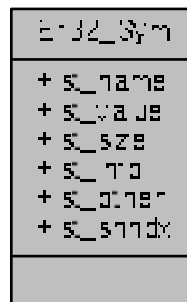
- [l4/util/elf.h](#)

14.54 Elf32_Sym Struct Reference

ELF32 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Sym:



Data Fields

- [Elf32_Word st_name](#)
name of symbol (idx symstrtab)
- [Elf32_Addr st_value](#)
value of associated symbol
- [Elf32_Word st_size](#)
size of associated symbol
- unsigned char [st_info](#)
type and binding info
- unsigned char [st_other](#)
undefined
- [Elf32_Half st_shndx](#)
associated section header

14.54.1 Detailed Description

ELF32 symbol table entry.

Definition at line 801 of file [elf.h](#).

The documentation for this struct was generated from the following file:

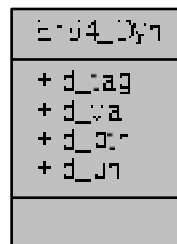
- [l4/util/elf.h](#)

14.55 Elf64_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Dyn:



Data Fields

- [Elf64_Sxword d_tag](#)
see DT_ values
- [Elf64_Xword d_val](#)
integer values with various interpret.
- [Elf64_Addr d_ptr](#)
program virtual addresses

14.55.1 Detailed Description

ELF64 dynamic entry.

Definition at line 509 of file [elf.h](#).

14.55.2 Field Documentation

14.55.2.1 d_val

`Elf64_Xword Elf64_Dyn::d_val`

integer values with various interpret.

Definition at line 512 of file [elf.h](#).

The documentation for this struct was generated from the following file:

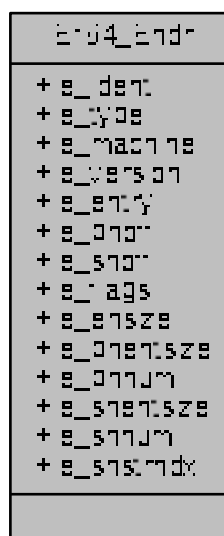
- [l4/util/elf.h](#)

14.56 Elf64_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Ehdr:



Data Fields

- [Elf64_Half e_type](#)
type of ELF file
- [Elf64_Half e_machine](#)
required architecture
- [Elf64_Word e_version](#)
file version
- [Elf64_Addr e_entry](#)
initial eip
- [Elf64_Off e_phoff](#)
offset of program header table
- [Elf64_Off e_shoff](#)
offset of file header table
- [Elf64_Word e_flags](#)
processor-specific flags
- [Elf64_Half e_ehsize](#)
size of ELF header
- [Elf64_Half e_phentsize](#)
size of program header entry
- [Elf64_Half e_phnum](#)

of entries in prog.

- [Elf64_Half e_shentsize](#)
size of section header entry
- [Elf64_Half e_shnum](#)

of entries in sect.

- [Elf64_Half e_shstrndx](#)
sect.head.tab.idx of strtb

14.56.1 Detailed Description

ELF64 header.

Definition at line 142 of file [elf.h](#).

14.56.2 Field Documentation

14.56.2.1 e_phnum

[Elf64_Half](#) [Elf64_Ehdr::e_phnum](#)

of entries in prog.

head. tab.

Definition at line 153 of file [elf.h](#).

14.56.2.2 e_shnum

[Elf64_Half](#) Elf64_Ehdr::e_shnum

of entries in sect.

head. tab.

Definition at line 155 of file [elf.h](#).

The documentation for this struct was generated from the following file:

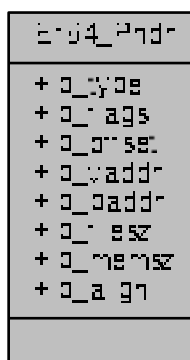
- [l4/util/elf.h](#)

14.57 Elf64_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Phdr:



Data Fields

- [Elf64_Word p_type](#)
type of program section
- [Elf64_Word p_flags](#)
flags
- [Elf64_Off p_offset](#)
file offset of program section
- [Elf64_Addr p_vaddr](#)
memory address of prog section
- [Elf64_Addr p_paddr](#)
physical address (ignored)
- [Elf64_Xword p_filesz](#)
file size of program section
- [Elf64_Xword p_memsz](#)
memory size of program section
- [Elf64_Xword p_align](#)
alignment of section

14.57.1 Detailed Description

ELF64 program header.

Definition at line 431 of file [elf.h](#).

The documentation for this struct was generated from the following file:

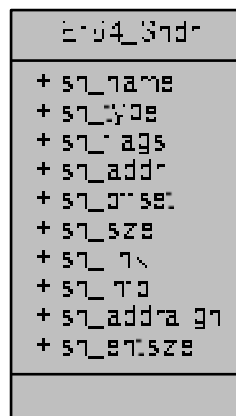
- [l4/util/elf.h](#)

14.58 Elf64_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Shdr:



Data Fields

- [Elf64_Word sh_name](#)
name of sect (idx into strtab)
- [Elf64_Word sh_type](#)
section's type
- [Elf64_Xword sh_flags](#)
section's flags
- [Elf64_Addr sh_addr](#)
memory address of section
- [Elf64_Off sh_offset](#)
file offset of section
- [Elf64_Xword sh_size](#)
file size of section
- [Elf64_Word sh_link](#)
idx to associated header section
- [Elf64_Word sh_info](#)
extra info of header section
- [Elf64_Xword sh_addralign](#)
address alignment constraints
- [Elf64_Xword sh_entsize](#)
size of entry if sect is table

14.58.1 Detailed Description

ELF64 section header.

Definition at line 356 of file [elf.h](#).

The documentation for this struct was generated from the following file:

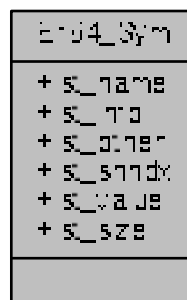
- [l4/util/elf.h](#)

14.59 Elf64_Sym Struct Reference

ELF64 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Sym:



Data Fields

- [Elf64_Word st_name](#)
name of symbol (idx symstrtab)
- unsigned char [st_info](#)
type and binding info
- unsigned char [st_other](#)
undefined
- [Elf64_Half st_shndx](#)
associated section header
- [Elf64_Addr st_value](#)
value of associated symbol
- [Elf64_Xword st_size](#)
size of associated symbol

14.59.1 Detailed Description

ELF64 symbol table entry.

Definition at line 811 of file [elf.h](#).

The documentation for this struct was generated from the following file:

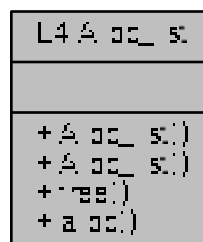
- [l4/util/elf.h](#)

14.60 L4::Alloc_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc_list:



14.60.1 Detailed Description

A simple list-based allocator.

Definition at line 31 of file [alloc.h](#).

The documentation for this class was generated from the following file:

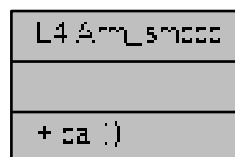
- [l4/cxx/alloc.h](#)

14.61 L4::Arm_smccc Class Reference

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

Inherits [L4::Kobject_0t](#)< Derived, PROTO, S_DEMAND >.

Collaboration diagram for [L4::Arm_smccc](#):



Public Member Functions

- [l4_msgtag_t](#) call ([l4_umword_t](#) func, [l4_umword_t](#) in0, [l4_umword_t](#) in1, [l4_umword_t](#) in2, [l4_umword_t](#) in3, [l4_umword_t](#) in4, [l4_umword_t](#) in5, [l4_umword_t](#) *out0, [l4_umword_t](#) *out1, [l4_umword_t](#) *out2, [l4_umword_t](#) *out3, [l4_umword_t](#) client_id)

ARM SMC/HVC function call.

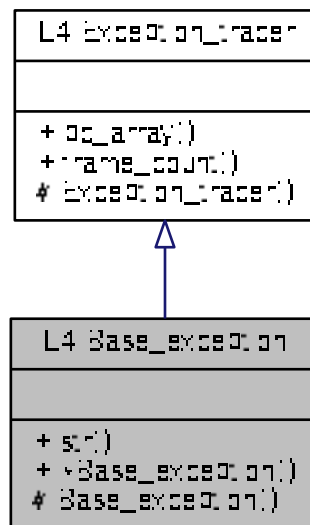
14.61.1 Detailed Description

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

Definition at line 20 of file [arm_smccc](#).

14.61.2 Member Function Documentation

Collaboration diagram for L4::Base_exception:



Public Member Functions

- virtual char const * [str](#) () const =0 throw ()
Return a human readable string for the exception.
- virtual [~Base_exception](#) () throw ()
Destruction.

Protected Member Functions

- [Base_exception](#) () throw ()
Create a base exception.

14.62.1 Detailed Description

Base class for all exceptions, thrown by the [L4Re](#) framework.

This is the abstract base of all exceptions thrown within the [L4Re](#) framework. It is basically also a good idea to use it as base of all user defined exceptions.

Definition at line [116](#) of file [exceptions](#).

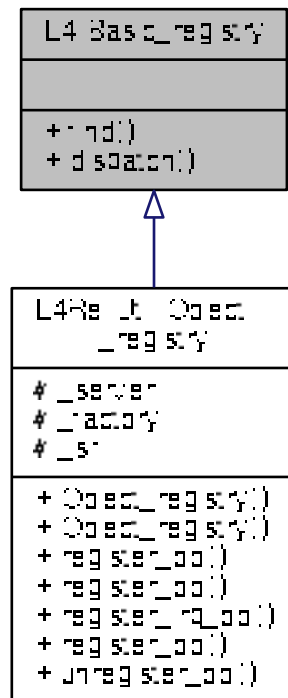
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

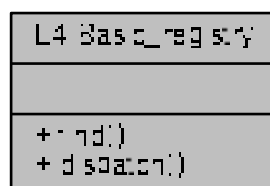
14.63 L4::Basic_registry Class Reference

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

Inheritance diagram for L4::Basic_registry:



Collaboration diagram for L4::Basic_registry:



Static Public Member Functions

- static Value * [find](#) (l4_umword_t label)

Get the server object for an [lpc_gate](#) label.

- static [l4_msgtag_t](#) dispatch ([l4_msgtag_t](#) tag, [l4_umword_t](#) label, [l4_utcb_t](#) *utcb)

The dispatch function called by the server loop.

14.63.1 Detailed Description

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

Definition at line 509 of file [ipc_epiface](#).

14.63.2 Member Function Documentation

14.63.2.1 dispatch()

```
static l4\_msgtag\_t L4::Basic_registry::dispatch (
    l4\_msgtag\_t tag,
    l4\_umword\_t label,
    l4\_utcb\_t * utcb ) [inline], [static]
```

The dispatch function called by the server loop.

This function forwards the message to the server object identified by the given *label*.

Parameters

<i>tag</i>	The message tag used for the invocation.
<i>label</i>	The label used to find the object including the rights bits of the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

Returns

The return code from the object's dispatch function or -L4_ENOENT if the object does not exist.

Definition at line 534 of file [ipc_epiface](#).

14.63.2.2 find()

```
static Value* L4::Basic_registry::find (
    l4\_umword\_t label ) [inline], [static]
```

Get the server object for an [lpc_gate](#) label.

Parameters

<i>label</i>	The label usually stored in an lpc_gate .
--------------	---

Returns

A pointer to the Epiface identified by the given label.

Definition at line [518](#) of file [ipc_epiface](#).

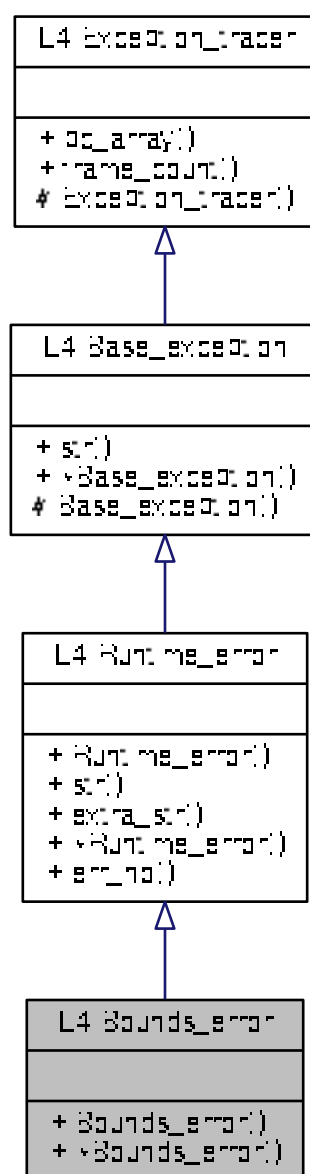
The documentation for this class was generated from the following file:

- `I4/sys/cxx/ipc_epiface`

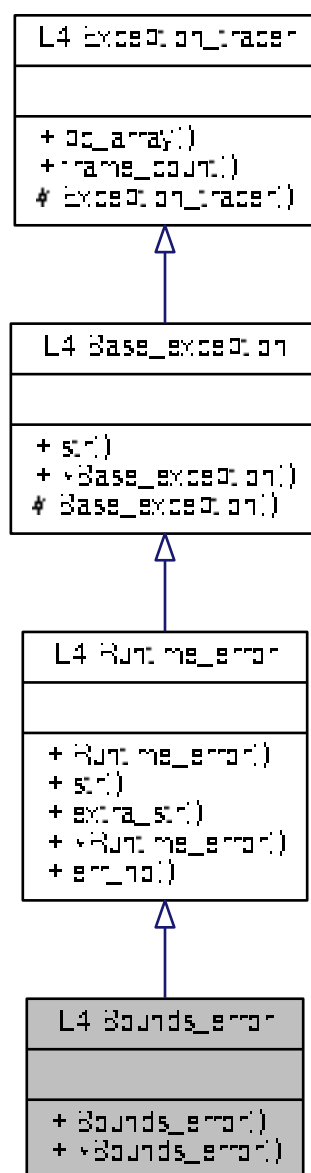
14.64 L4::Bounds_error Class Reference

Access out of bounds.

Inheritance diagram for L4::Bounds_error:



Collaboration diagram for L4::Bounds_error:



Additional Inherited Members

14.64.1 Detailed Description

Access out of bounds.

Definition at line 289 of file [exceptions](#).

The documentation for this class was generated from the following file:

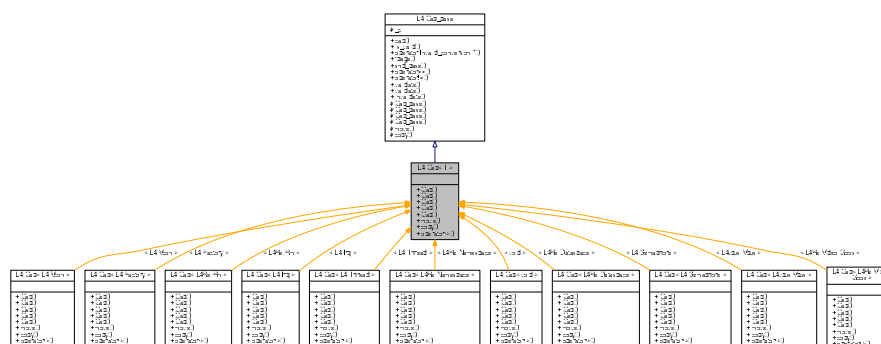
- [l4/cxx/exceptions](#)

14.65 L4::Cap< T > Class Template Reference

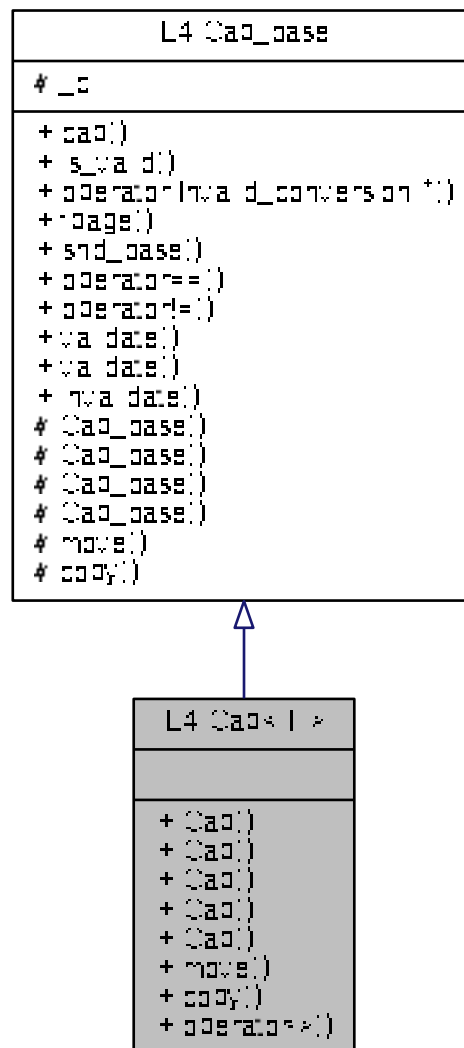
C++ interface for capabilities.

```
#include <capability.h>
```

Inheritance diagram for L4::Cap< T >:



Collaboration diagram for L4::Cap< T >:



Public Member Functions

- `template<typename O >`
`Cap (Cap< O > const &o) throw ()`
Create a copy from o, supporting implicit type casting.
- `Cap (Cap_type cap) throw ()`
Constructor to create an invalid capability selector.
- `Cap (l4_default_caps_t cap) throw ()`
Initialize capability with one of the default capability selectors.
- `Cap (l4_cap_idx_t idx=L4_INVALID_CAP) throw ()`
Initialize capability, defaults to the invalid capability selector.
- `Cap (No_init_type) throw ()`

Create an uninitialized cap selector.

- [Cap move](#) ([Cap](#) const &src) const

Move a capability to this cap slot.

- [Cap copy](#) ([Cap](#) const &src) const

Copy a capability to this cap slot.

- T * [operator->](#) () const throw ()

Member access of a T.

Friends

- class [L4::Kobject](#)

Additional Inherited Members

14.65.1 Detailed Description

```
template<typename T>
class L4::Cap< T >
```

C++ interface for capabilities.

Template Parameters

T	Type of the object the capability points to.
-------------------	--

The C++ version of a capability is comparable to a pointer, in fact it is a kind of smart pointer for our kernel objects and the objects derived from the kernel objects ([L4::Kobject](#)).

Add

```
#include <l4/sys/capability>
```

to your code to use the capability interface.

Examples:

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).↔

Definition at line 13 of file [capability.h](#).

14.65.2 Constructor & Destructor Documentation

14.65.2.1 [Cap\(\)](#) [1/4]

```
template<typename T>
template<typename O >
L4::Cap< T >::Cap (
    Cap< O > const & o ) throw ()    [inline]
```

Create a copy from o, supporting implicit type casting.

Parameters

<i>o</i>	The source selector that shall be copied (and casted).
----------	--

Definition at line 244 of file [capability.h](#).

14.65.2.2 **Cap()** [2/4]

```
template<typename T>
L4::Cap< T >::Cap (
    Cap_type cap ) throw )    [inline]
```

Constructor to create an invalid capability selector.

Parameters

<i>cap</i>	Capability selector.
------------	----------------------

Definition at line 251 of file [capability.h](#).

14.65.2.3 **Cap()** [3/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_default_caps_t cap ) throw )    [inline]
```

Initialize capability with one of the default capability selectors.

Parameters

<i>cap</i>	Capability selector.
------------	----------------------

Definition at line 257 of file [capability.h](#).

14.65.2.4 **Cap()** [4/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_cap_idx_t idx = L4_INVALID_CAP ) throw )    [inline], [explicit]
```

Initialize capability, defaults to the invalid capability selector.

Parameters

<i>idx</i>	Capability selector.
------------	----------------------

Definition at line 263 of file [capability.h](#).

14.65.3 Member Function Documentation

14.65.3.1 copy()

```
template<typename T>
Cap L4::Cap< T >::copy (
    Cap< T > const & src ) const [inline]
```

Copy a capability to this cap slot.

Parameters

<i>src</i>	the source capability slot.
------------	-----------------------------

Definition at line 286 of file [capability.h](#).

14.65.3.2 move()

```
template<typename T>
Cap L4::Cap< T >::move (
    Cap< T > const & src ) const [inline]
```

Move a capability to this cap slot.

Parameters

<i>src</i>	the source capability slot.
------------	-----------------------------

After this operation the source slot is no longer valid.

Definition at line 276 of file [capability.h](#).

The documentation for this class was generated from the following file:

- I4/sys/cxx/capability.h

Public Member Functions

- [l4_cap_idx_t cap](#) () const throw ()
Return capability selector.
- bool [is_valid](#) () const throw ()
Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).
- [l4_fpage_t fpage](#) (unsigned rights=[L4_FPAGE_RWX](#)) const throw ()
Return flex-page for the capability.
- [l4_umword_t snd_base](#) (unsigned grant=0, [l4_cap_idx_t](#) base=[L4_INVALID_CAP](#)) const throw ()
Return send base.
- bool [operator==](#) ([Cap_base](#) const &o) const throw ()
Test if two capabilities are equal.
- bool [operator!=](#) ([Cap_base](#) const &o) const throw ()
Test if two capabilities are not equal.
- [l4_msgtag_t validate](#) ([l4_utcb_t](#) *u=[l4_utcb\(\)](#)) const throw ()
Check whether a capability is present (refers to an object).
- [l4_msgtag_t validate](#) ([Cap](#)< [Task](#) > task, [l4_utcb_t](#) *u=[l4_utcb\(\)](#)) const throw ()
Check whether a capability is present (refers to an object).
- void [invalidate](#) () throw ()
Set this capability to invalid (L4_INVALID_CAP).

Protected Member Functions

- [Cap_base](#) ([l4_cap_idx_t](#) c) throw ()
Generate a capability from its C representation.
- [Cap_base](#) ([Cap_type](#) cap) throw ()
Constructor to create an invalid capability.
- [Cap_base](#) ([l4_default_caps_t](#) cap) throw ()
Initialize capability with one of the default capabilities.
- [Cap_base](#) () throw ()
Create an uninitialized instance.
- void [move](#) ([Cap_base](#) const &src) const
Replace this capability with the contents of src.
- void [copy](#) ([Cap_base](#) const &src) const
Copy a capability.

Protected Attributes

- [l4_cap_idx_t _c](#)
The C representation of a capability selector.

14.66.1 Detailed Description

Base class for all kinds of capabilities.

Attention

This class is not for direct use, use [L4::Cap](#) instead.

This class contains all the things that are independent of the type of the object referred by the capability.

See also

[L4::Cap](#) for typed capabilities.

Definition at line 25 of file [capability.h](#).

14.66.2 Member Enumeration Documentation

14.66.2.1 Cap_type

enum [L4::Cap_base::Cap_type](#)

Invalid capability type.

Enumerator

Invalid	Invalid capability selector.
---------	------------------------------

Definition at line [43](#) of file [capability.h](#).

14.66.2.2 No_init_type

enum [L4::Cap_base::No_init_type](#)

Special value for uninitialized capability objects.

Enumerator

No_init	Special value for constructing uninitialized Cap objects.
---------	---

Definition at line [32](#) of file [capability.h](#).

14.66.3 Constructor & Destructor Documentation

14.66.3.1 Cap_base() [1/2]

```
L4::Cap_base::Cap_base (
    l4\_cap\_idx\_t c ) throw ()    [inline], [explicit], [protected]
```

Generate a capability from its C representation.

Parameters

<i>c</i>	The C capability
----------	------------------

Definition at line [144](#) of file [capability.h](#).

14.66.3.2 Cap_base() [2/2]

```
L4::Cap_base::Cap_base (
    l4_default_caps_t cap ) throw ()    [inline], [explicit], [protected]
```

Initialize capability with one of the default capabilities.

Parameters

<i>cap</i>	Capability.
------------	-------------

Definition at line 155 of file [capability.h](#).

14.66.4 Member Function Documentation

14.66.4.1 cap()

```
l4_cap_idx_t L4::Cap_base::cap ( ) const throw ()    [inline]
```

Return capability selector.

Returns

Capability selector.

Definition at line 52 of file [capability.h](#).

Referenced by [L4Re::Util::Cap_alloc_base::alloc\(\)](#), [L4vbus::Vbus::assign_dma_domain\(\)](#), [L4::lcu::bind\(\)](#), [L4::cap←_cast\(\)](#), [L4::cap_reinterpret_cast\(\)](#), [L4::Irq_mux::chain\(\)](#), [L4Re::Util::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4Re::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4::Factory::create_factory\(\)](#), [L4::Factory::create_gate\(\)](#), [L4←::Factory::create_irq\(\)](#), [L4::Factory::create_task\(\)](#), [L4::Factory::create_thread\(\)](#), [L4::Factory::create_vm\(\)](#), [L4Re::←Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4::Invalid_capability::Invalid_capability\(\)](#), [L4virtio::Svr::Device_t< Ds_data >::mem_info\(\)](#), [L4::Smart_cap< T, SMART >::operator->\(\)](#), [L4::Cap< L4Re::Video::Goos >::operator->\(\)](#), [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#), [L4::lpc_svr::Server_iface::rcv_cap\(\)](#), [L4::←Smart_cap< T, SMART >::Smart_cap\(\)](#), [L4::lcu::unbind\(\)](#), and [L4vbus::lcu::vicu\(\)](#).

Parameters

<code>src</code>	the source capability.
------------------	------------------------

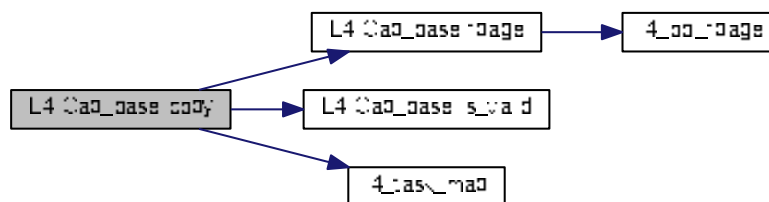
After this operation this capability refers to the same object as `src`.

Definition at line 187 of file `capability.h`.

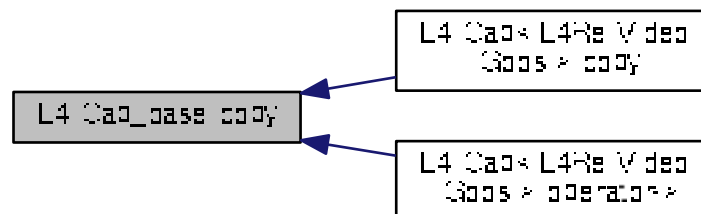
References `fpage()`, `is_valid()`, `L4_BASE_TASK_CAP`, `L4_CAP_FPAGE_RWSD`, and `l4_task_map()`.

Referenced by `L4::Cap< L4Re::Video::Goos >::copy()`, and `L4::Cap< L4Re::Video::Goos >::operator->()`.

Here is the call graph for this function:



Here is the caller graph for this function:



14.66.4.3 fpage()

```

l4_fpage_t L4::Cap_base::fpage (
    unsigned rights = L4_FPAGE_RWX ) const throw ()    [inline]

```

Return flex-page for the capability.

Parameters

<i>rights</i>	Rights, defaults to 'rwx'
---------------	---------------------------

Returns

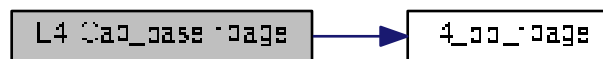
flex-page

Definition at line 72 of file [capability.h](#).

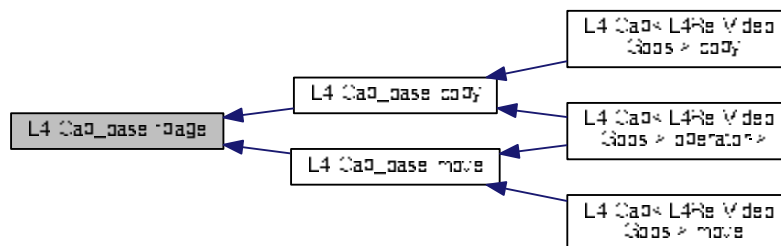
References [l4_obj_fpage\(\)](#).

Referenced by [copy\(\)](#), and [move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.66.4.4 is_valid()

```
bool L4::Cap_base::is_valid ( ) const throw ( ) [inline]
```

Test whether the capability is a valid capability index (i.e., not `L4_INVALID_CAP`).

Returns

True if capability is not invalid, false if invalid

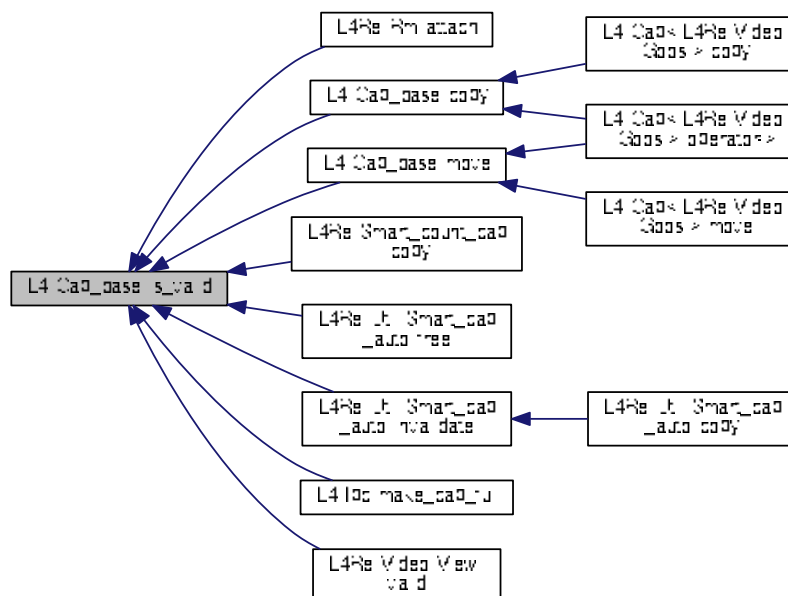
Examples:

[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), and [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 60 of file [capability.h](#).

Referenced by [L4Re::Rm::attach\(\)](#), [copy\(\)](#), [L4Re::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::invalidate\(\)](#), [L4::lpc::make_cap_full\(\)](#), [move\(\)](#), and [L4Re::Video::View::valid\(\)](#).

Here is the caller graph for this function:

**14.66.4.5 move()**

```
void L4::Cap_base::move (
    Cap_base const & src ) const [inline], [protected]
```

Replace this capability with the contents of `src`.

Parameters

<code>src</code>	the source capability.
------------------	------------------------

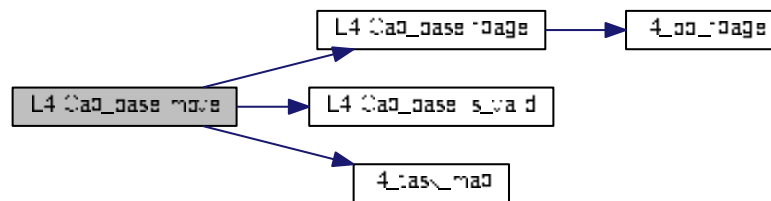
After the operation this capability refers to the object formerly referred to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 171 of file [capability.h](#).

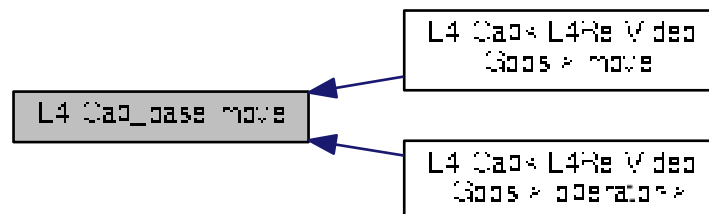
References [fpage\(\)](#), [is_valid\(\)](#), [L4_BASE_TASK_CAP](#), [L4_CAP_FPAGE_RWSD](#), [L4_MAP_ITEM_GRANT](#), and [l4_task_map\(\)](#).

Referenced by [L4::Cap< L4Re::Video::Goos >::move\(\)](#), and [L4::Cap< L4Re::Video::Goos >::operator->\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.66.4.6 `snd_base()`

```

l4_umword_t L4::Cap_base::snd_base (
    unsigned grant = 0,
    l4_cap_idx_t base = L4_INVALID_CAP ) const throw ()    [inline]
  
```

Return send base.

Parameters

<i>grant</i>	True object should be granted.
<i>base</i>	Base capability (first in a bundle of aligned capabilities)

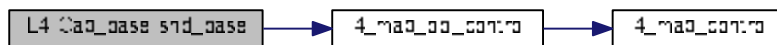
Returns

Map object.

Definition at line 83 of file [capability.h](#).

References [L4_INVALID_CAP](#), and [l4_map_obj_control\(\)](#).

Here is the call graph for this function:



14.66.4.7 validate() [1/2]

```
l4_msgtag_t L4::Cap_base::validate (
    l4_utcb_t * u = l4_utcb() ) const throw()    [inline]
```

Check whether a capability is present (refers to an object).

Parameters

<i>u</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
----------	---

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line 90 of file [capability](#).

14.66.4.8 validate() [2/2]

```
l4_msgtag_t L4::Cap_base::validate (
    Cap< Task > task,
    l4_utcb_t * u = l4_utcb() ) const throw ()    [inline]
```

Check whether a capability is present (refers to an object).

Parameters

<i>task</i>	Task to check the capability in.
<i>u</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line [83](#) of file [capability](#).

14.66.5 Field Documentation**14.66.5.1 _c**

```
l4_cap_idx_t L4::Cap_base::_c    [protected]
```

The C representation of a capability selector.

Definition at line [198](#) of file [capability.h](#).

Referenced by [L4::Smart_cap< T, SMART >::operator->\(\)](#), and [L4::Smart_cap< T, SMART >::Smart_cap\(\)](#).

The documentation for this class was generated from the following files:

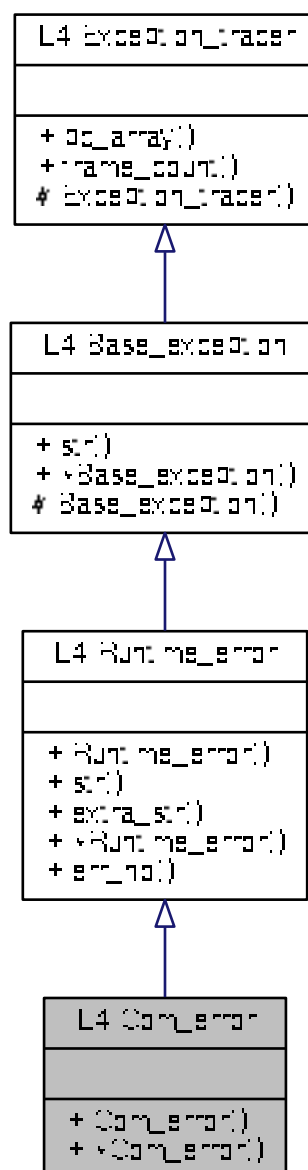
- [l4/sys/cxx/capability.h](#)
- [l4/sys/capability](#)

14.67 L4::Com_error Class Reference

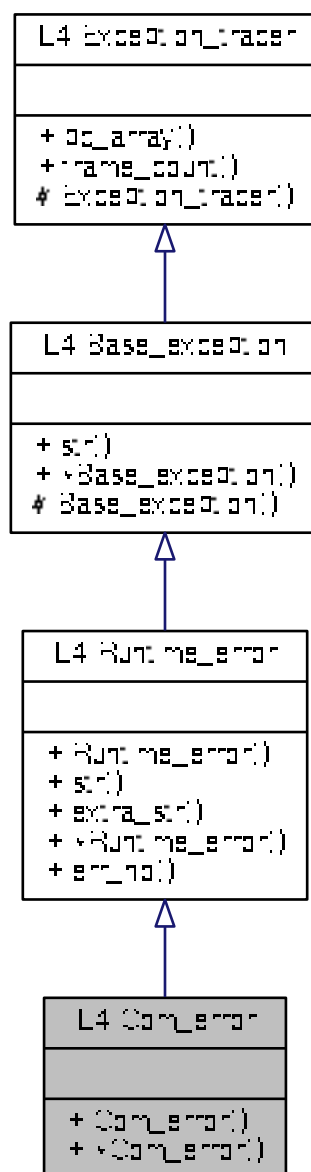
Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com_error:



Collaboration diagram for L4::Com_error:



Public Member Functions

- [Com_error](#) (long err) throw ()

Create a [Com_error](#) for the given [L4](#) IPC error code.

Additional Inherited Members

14.67.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of [L4 IPC](#).

Definition at line [274](#) of file [exceptions](#).

14.67.2 Constructor & Destructor Documentation

14.67.2.1 Com_error()

```
L4::Com_error::Com_error (
    long err ) throw ()    [inline], [explicit]
```

Create a [Com_error](#) for the given [L4](#) IPC error code.

Parameters

<i>err</i>	The L4 IPC error code (l4_ipc... return value).
------------	---

Definition at line [281](#) of file [exceptions](#).

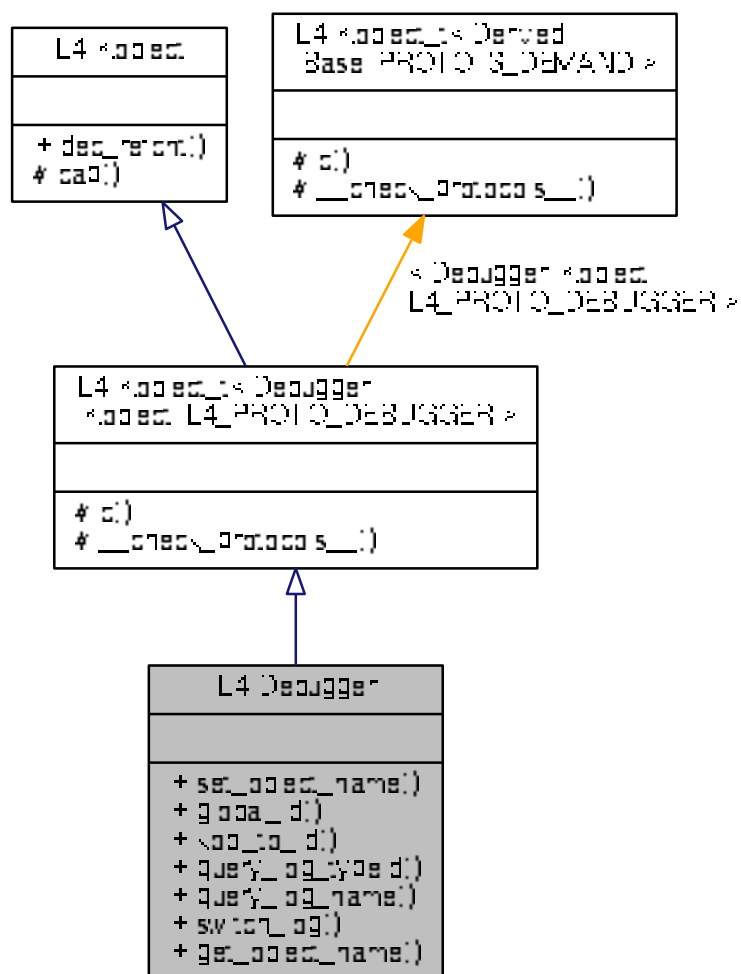
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

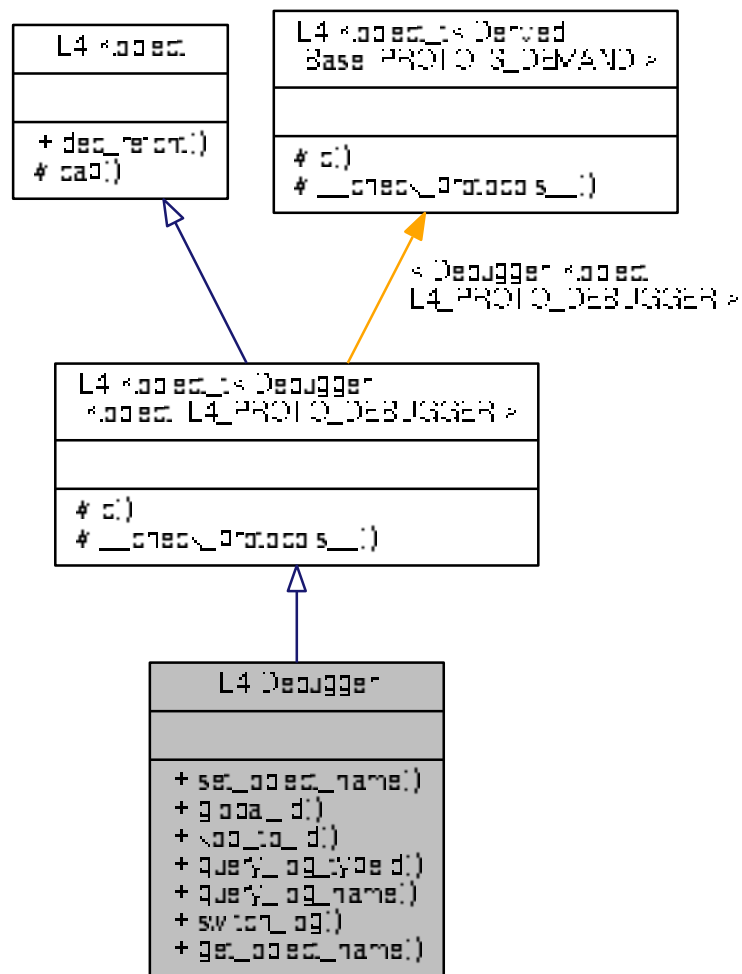
14.68 L4::Debugger Class Reference

C++ kernel debugger API.

Inheritance diagram for L4::Debugger:



Collaboration diagram for L4::Debugger:



Public Member Functions

- [l4_msgtag_t set_object_name](#) (const char *name, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Set the name of a kernel object.
- unsigned long [global_id](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Get the globally unique ID of the object behind a capability.
- unsigned long [kobj_to_id](#) ([l4_addr_t](#) kobjp, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Get the globally unique ID of the object behind the kobject pointer.
- int [query_log_typeid](#) (const char *name, unsigned idx, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Query the log-id for a log type.
- int [query_log_name](#) (unsigned idx, char *name, unsigned namelen, char *shortname, unsigned shortname-len, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Query the name of a log type given the ID.
- [l4_msgtag_t switch_log](#) (const char *name, unsigned on_off, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()

Set or unset log.

- `l4_msgtag_t get_object_name` (unsigned id, char *name, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) throw ()

Get name of object with Id id.

Additional Inherited Members

14.68.1 Detailed Description

C++ kernel debugger API.

Attention

This API is subject to change! Do not rely on it in production code.

This API is to be used for debugging exclusively.

This is the API for accessing kernel-debugger functionality from user-level programs. Specifically, it provides functionality to enrich the kernel debugger with insights into the program. The purpose is to facilitate debugging with the kernel debugger. For instance, a developer might choose to name the threads of her program so that she can find them in the kernel debugger thread list.

This API interacts with a kernel object that interfaces with the kernel debugger, the jdb-kernel object. The jdb-kernel object is fix and only available when the kernel debugger is built into the microkernel. The developer needs to pass the capability through to her program.

Include File

```
#include <l4/sys/debugger>
```

Definition at line 53 of file `debugger`.

14.68.2 Member Function Documentation

14.68.2.1 get_object_name()

```
l4_msgtag_t L4::Debugger::get_object_name (
    unsigned id,
    char * name,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get name of object with Id id.

Parameters

	<i>id</i>	Id of the object whose name is asked.
out	<i>name</i>	Buffer to copy the name into. The buffer must be allocated by the caller.
	<i>size</i>	Length of the <code>name</code> buffer.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Definition at line 159 of file [debugger](#).

14.68.2.2 global_id()

```
unsigned long L4::Debugger::global_id (  
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get the globally unique ID of the object behind a capability.

Parameters

<i>utcb</i>	The UTCB to use for the operation.
-------------	------------------------------------

Return values

$\sim 0UL$	The capability is invalid.
≥ 0	The global debugger id.

Definition at line 82 of file [debugger](#).

14.68.2.3 kobj_to_id()

```
unsigned long L4::Debugger::kobj_to_id (  
    l4_addr_t kobjp,  
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

<i>kobjp</i>	Kobject pointer
<i>utcb</i>	The UTCB to use for the operation.

Return values

$\sim 0UL$	The capability or the Kobject pointer are invalid.
≥ 0	The globally unique id.

Definition at line 94 of file [debugger](#).

14.68.2.4 `query_log_name()`

```
int L4::Debugger::query_log_name (
    unsigned idx,
    char * name,
    unsigned namelen,
    char * shortname,
    unsigned shortnamelen,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Query the name of a log type given the ID.

Parameters

	<i>idx</i>	ID to query.
out	<i>name</i>	Buffer to copy name to. The buffer must be allocated by the caller.
	<i>namelen</i>	Buffer length of name.
out	<i>shortname</i>	Buffer to copy <i>shortname</i> to. The buffer must be allocated by the caller.
	<i>shortnamelen</i>	Buffer length of <i>shortname</i> .
	<i>utcb</i>	The UTCB to use for the operation.

Return values

0	Success
<0	Error

Definition at line 127 of file [debugger](#).

14.68.2.5 `query_log_typeid()`

```
int L4::Debugger::query_log_typeid (
    const char * name,
    unsigned idx,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Query the log-id for a log type.

Parameters

<i>name</i>	Name to query for.
<i>idx</i>	Idx to start searching, start with 0
<i>utcb</i>	The UTCB to use for the operation.

Return values

>=0	Id
<0	Error

Definition at line 108 of file [debugger](#).

14.68.2.6 set_object_name()

```
l4_msgtag_t L4::Debugger::set_object_name (
    const char * name,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Set the name of a kernel object.

Parameters

<i>name</i>	Name
<i>utcb</i>	The UTCB to use for the operation.

Returns

System call return tag.

Definition at line 70 of file [debugger](#).

14.68.2.7 switch_log()

```
l4_msgtag_t L4::Debugger::switch_log (
    const char * name,
    unsigned on_off,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Set or unset log.

Parameters

<i>name</i>	Name of the log type.
<i>on_off</i>	1: turn log on, 0: turn log off
<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Definition at line 144 of file [debugger](#).

The documentation for this class was generated from the following file:

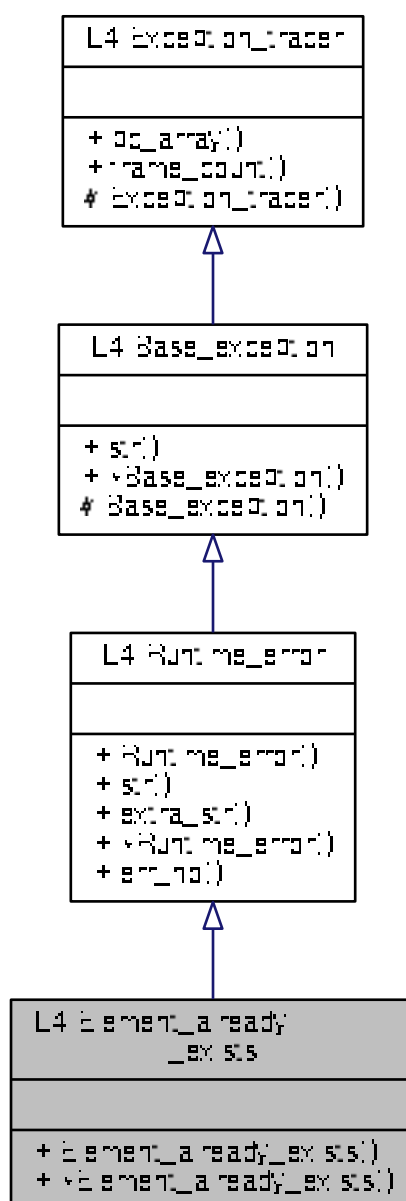
- [l4/sys/debugger](#)

14.69 L4::Element_already_exists Class Reference

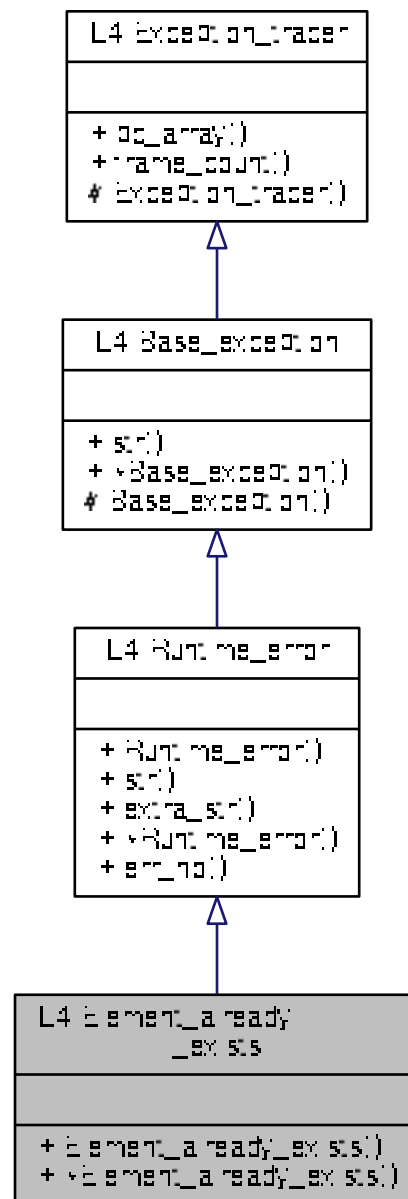
[Exception](#) for duplicate element insertions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_already_exists:



Collaboration diagram for L4::Element_already_exists:



Additional Inherited Members

14.69.1 Detailed Description

[Exception](#) for duplicate element insertions.

Definition at line 203 of file [exceptions](#).

The documentation for this class was generated from the following file:

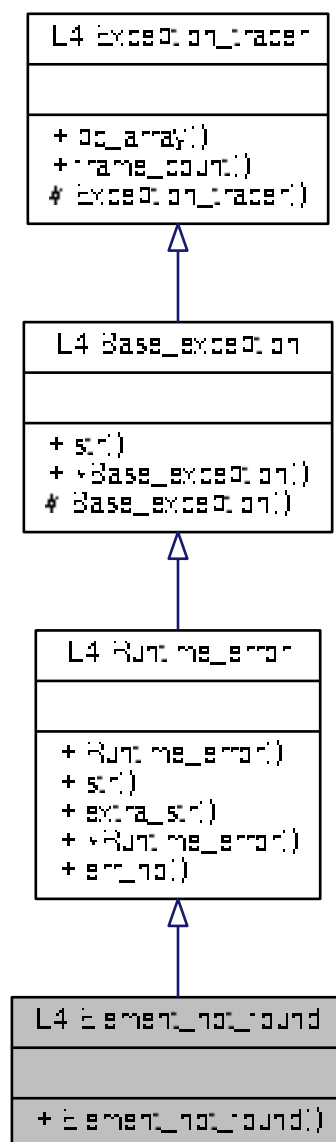
- [l4/cxx/exceptions](#)

14.70 L4::Element_not_found Class Reference

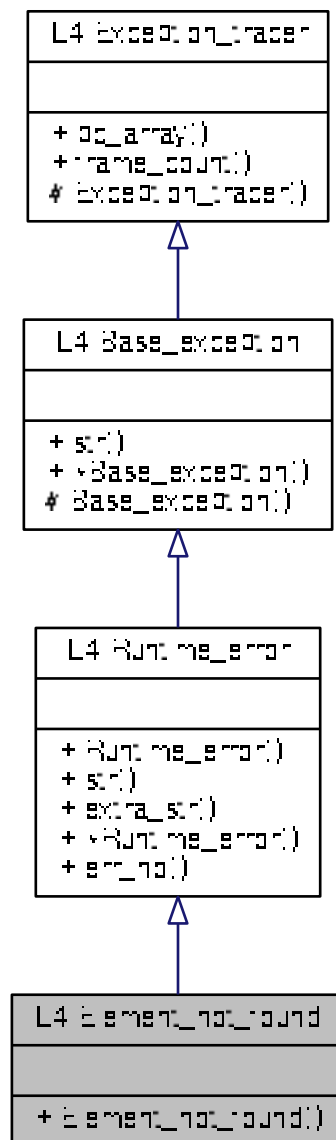
[Exception](#) for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_not_found:



Collaboration diagram for L4::Element_not_found:



Additional Inherited Members

14.70.1 Detailed Description

Exception for a failed lookup (element not found).

Definition at line 231 of file [exceptions](#).

The documentation for this class was generated from the following file:

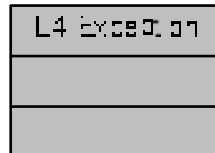
- [l4/cxx/exceptions](#)

14.71 L4::Exception Class Reference

[Exception](#) interface.

Inherits [L4::Kobject_0t](#)< Derived, PROTO, S_DEMAND >.

Collaboration diagram for L4::Exception:



Public Types

- typedef [L4::Typeid::Rpc_nocode](#)< exception_t > [Rpc](#)s
[Exception](#) call.

14.71.1 Detailed Description

[Exception](#) interface.

This class defines the interface for handling exception IPC. When an exception occurs during program execution, for example due to a division by zero, the kernel will synthesise an exception IPC and send it to the thread's exception handler, who can then handle it.

The exception handler is set with the [L4::Thread::control](#) interface.

Definition at line 42 of file [exception](#).

14.71.2 Member Typedef Documentation

14.71.2.1 Rpc

typedef [L4::Typeid::Rpc_nocode](#)<exception_t> [L4::Exception::Rpc](#)s

[Exception](#) call.

Parameters

	<i>regs</i>	Register state of the faulting thread.
	<i>rwin</i>	Receive window in the address space.
out	<i>fp</i>	Optional flex-page to resolve the exception.

Returns

Message tag containing error code.

Definition at line 60 of file [exception](#).

The documentation for this class was generated from the following file:

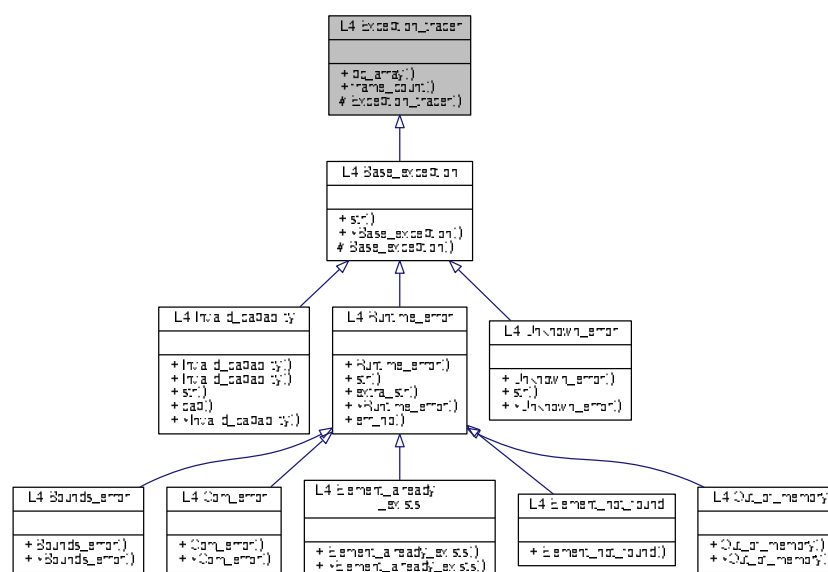
- [l4/sys/exception](#)

14.72 L4::Exception_tracer Class Reference

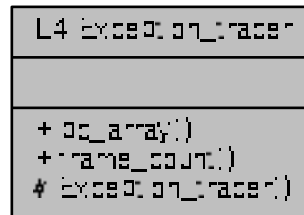
Back-trace support for exceptions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Exception_tracer:



Collaboration diagram for L4::Exception_tracer:



Public Member Functions

- void const *const * [pc_array](#) () const throw ()
Get the array containing the call trace.
- int [frame_count](#) () const throw ()
Get the number of entries that are valid in the call trace.

Protected Member Functions

- [Exception_tracer](#) () throw ()
Create a back trace.

14.72.1 Detailed Description

Back-trace support for exceptions.

This class holds an array of at most [L4_CXX_EXCEPTION_BACKTRACE](#) instruction pointers containing the call trace at the instant when an exception was thrown.

Definition at line 62 of file [exceptions](#).

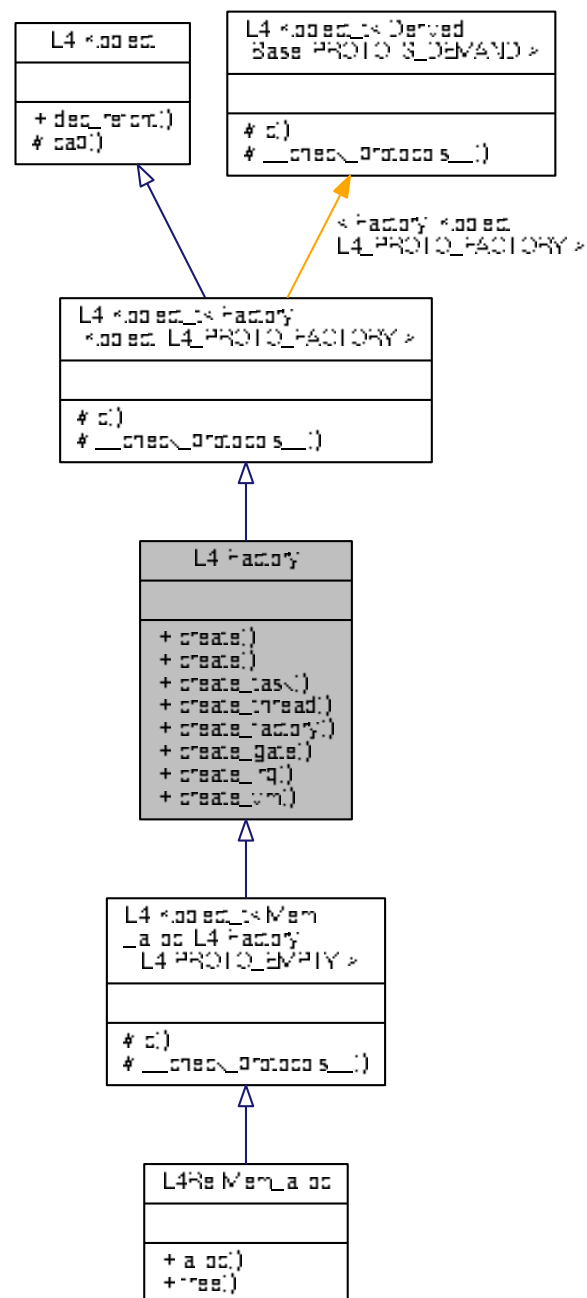
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

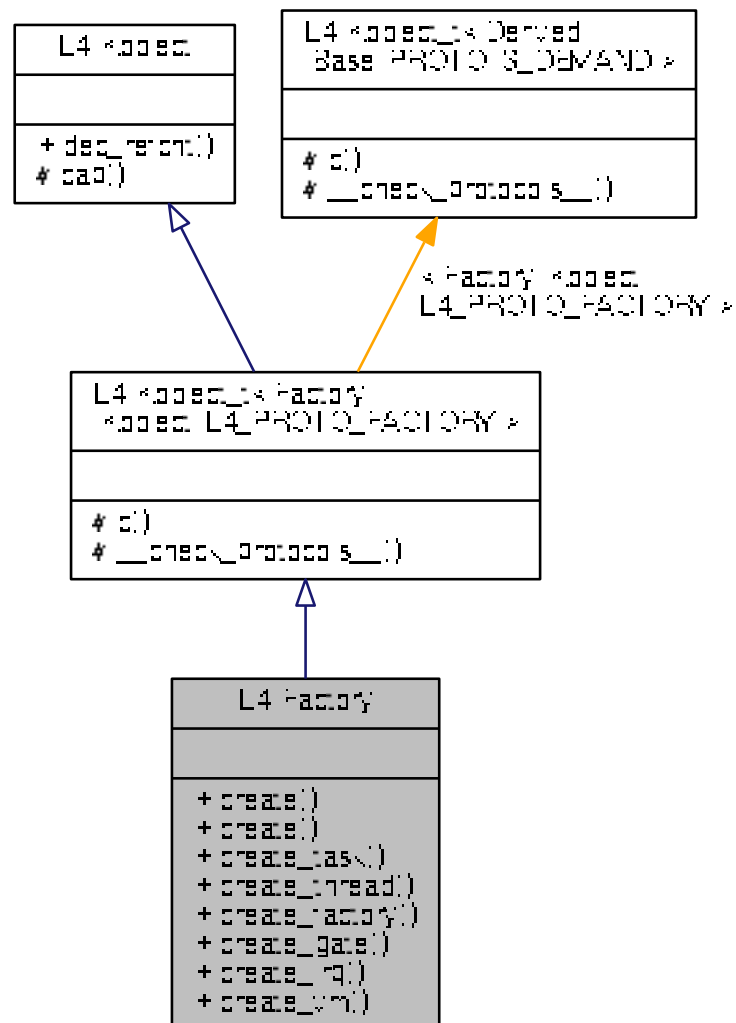
14.73 L4::Factory Class Reference

C++ [Factory](#) interface to create kernel objects.

Inheritance diagram for L4::Factory:



Collaboration diagram for L4::Factory:



Data Structures

- struct [Lstr](#)
Special type to add a pascal string into the factory create stream.
- struct [Nil](#)
Special type to add a void argument into the factory create stream.
- class [S](#)
Stream class for the [create\(\)](#) argument stream.

Public Member Functions

- [S create](#) ([Cap](#)< void > target, long obj, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()

Generic create call to the factory.

- `template<typename OBJ >
S create (Cap< OBJ > target, l4_utcb_t *utcb=l4_utcb()) throw ()`

Create call for typed capabilities.

- `l4_msgtag_t create_task (Cap< Task > const &target_cap, l4_fpage_t const &utcb_area, l4_utcb_t *utcb=l4_utcb()) throw ()`

Create a new task.

- `l4_msgtag_t create_thread (Cap< Thread > const &target_cap, l4_utcb_t *utcb=l4_utcb()) throw ()`

Create a new thread.

- `l4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, l4_utcb_t *utcb=l4_utcb()) throw ()`

Create a new factory.

- `l4_msgtag_t create_gate (Cap< void > const &target_cap, Cap< Thread > const &thread_cap, l4_umword_t label, l4_utcb_t *utcb=l4_utcb()) throw ()`

Create a new IPC gate.

- `l4_msgtag_t create_irq (Cap< Irq > const &target_cap, l4_utcb_t *utcb=l4_utcb()) throw ()`

Create a new IRQ.

- `l4_msgtag_t create_vm (Cap< Vm > const &target_cap, l4_utcb_t *utcb=l4_utcb()) throw ()`

Create a new virtual machine.

Additional Inherited Members

14.73.1 Detailed Description

C++ [Factory](#) interface to create kernel objects.

A factory is used to create all kinds of kernel objects:

- [L4::Task](#)
- [L4::Thread](#)
- [L4::Factory](#)
- [L4::lpc_gate](#)
- [L4::Irq](#)
- [L4::Vm](#)

The factory is equipped with a limit that limits the amount of kernel memory available to that factory.

Note

The limit does not give any guarantee for the amount of available kernel memory.

Include File

```
#include <l4/sys/factory>
```

For the C interface refer to [Factory](#).

Definition at line 55 of file [factory](#).

14.73.2 Member Function Documentation

14.73.2.1 `create()` [1/2]

```
S L4::Factory::create (
    Cap< void > target,
    long obj,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Generic create call to the factory.

Parameters

out	<i>target</i>	Capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new objects's capability into this slot.
	<i>obj</i>	The protocol ID that specifies which kind of object shall be created.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

A create stream that allows adding additional arguments to the `create()` call.

This method does currently not directly invoke the factory. It returns a stream that shall invoke the factory after adding all additional arguments.

Usage:

```
L4::Cap<L4Re::Namespace> ns = L4Re::Util::cap_alloc.
    alloc<L4Re::Namespace>();
factory->create(ns, L4Re::Namespace::Protocol) << "Argument text";
```

Definition at line 259 of file [factory](#).

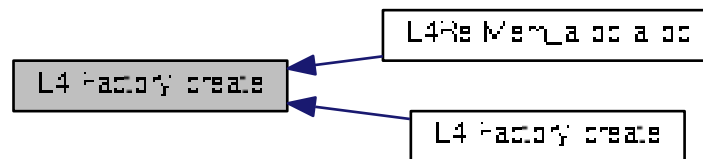
References [L4::Kobject::cap\(\)](#).

Referenced by [L4Re::Mem_alloc::alloc\(\)](#), and [create\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.73.2.2 create() [2/2]

```

template<typename OBJ >
S L4::Factory::create (
    Cap< OBJ > target,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

Create call for typed capabilities.

Template Parameters

<i>OBJ</i>	Capability type of the object to be created.
------------	--

Parameters

out	<i>target</i>	Capability of type OBJ.
	<i>utcb</i>	UTCB to use.

Returns

Argument stream to call the factory.

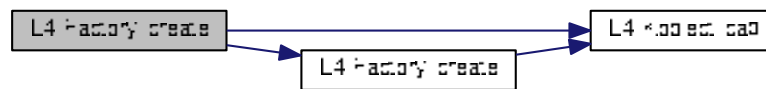
See also

[create](#)

Definition at line 276 of file [factory](#).

References [L4::Kobject::cap\(\)](#), [create\(\)](#), and [L4_INLINE_RPC_NF](#).

Here is the call graph for this function:



14.73.2.3 create_factory()

```

l4_msgtag_t L4::Factory::create_factory (
    Cap< Factory > const & target_cap,
    unsigned long limit,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Create a new factory.

Parameters

out	<i>target_cap</i>	The kernel stores the new factory's capability into this slot.
	<i>limit</i>	Limit for the new factory in bytes.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

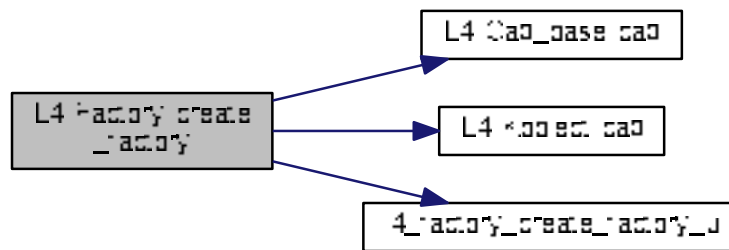
Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 342 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_factory_u\(\)](#).

Here is the call graph for this function:



14.73.2.4 create_gate()

```

l4_msgtag_t L4::Factory::create_gate (
    Cap< void > const & target_cap,
    Cap< Thread > const & thread_cap,
    l4_umword_t label,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Create a new IPC gate.

Parameters

out	<i>target_cap</i>	The kernel stores the new IPC gate's capability into this slot.
	<i>thread_cap</i>	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag containing one of the following return codes.

Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_ENOMEM</i>	Out-of-memory during allocation of the lpc_gate object.
<i>-L4_ENOENT</i>	<i>thread_cap</i> is void or points to something that is not a thread.
<i>-L4_EPERM</i>	No write rights on <i>thread_cap</i> .

An unbound IPC gate can be bound to a thread using [L4::lpc_gate::bind_thread\(\)](#).

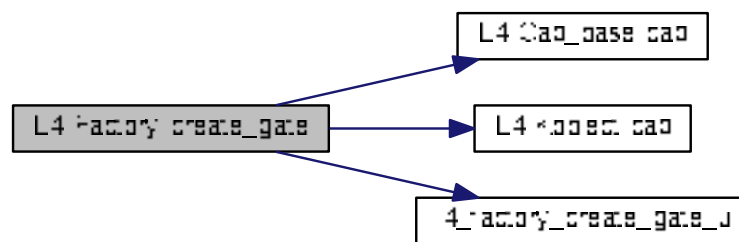
See also

[L4::lpc_gate](#)

Definition at line 372 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_gate_u\(\)](#).

Here is the call graph for this function:



14.73.2.5 create_irq()

```

l4_msgtag_t L4::Factory::create_irq (
    Cap< Irq >const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

Create a new IRQ.

Parameters

out	<i>target_cap</i>	The kernel stores the new IRQ's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Irq>` as argument instead.

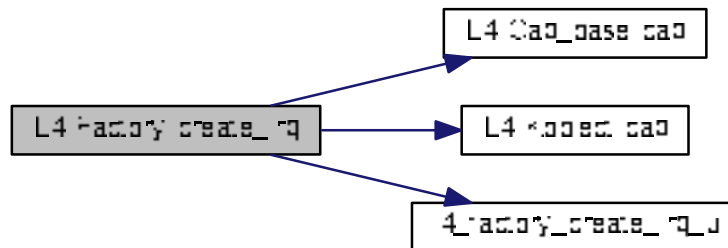
See also

[L4::Irq](#)

Definition at line 390 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_irq_u\(\)](#).

Here is the call graph for this function:



14.73.2.6 create_task()

```

l4_msgtag_t L4::Factory::create_task (
    Cap< Task > const & target_cap,
    l4_fpage_t const & utcb_area,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Create a new task.

Parameters

out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
	<i>utcb_area</i>	Flexpage that describes an area in the address space of the new task, where the kernel should map the kernel-allocated kernel-user memory to. The kernel uses the kernel-user memory to store UTCBs and vCPU state-save-areas of the new task.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

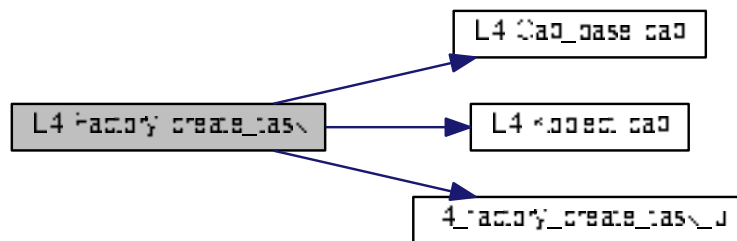
See also

[L4::Task](#)

Definition at line 306 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_task_u\(\)](#).

Here is the call graph for this function:



14.73.2.7 create_thread()

```

l4_msgtag_t L4::Factory::create_thread (
    Cap< Thread > const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

Create a new thread.

Parameters

out	<i>target_cap</i>	The kernel stores the new thread's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use [create\(\)](#) with [Cap<Thread>](#) as argument instead.

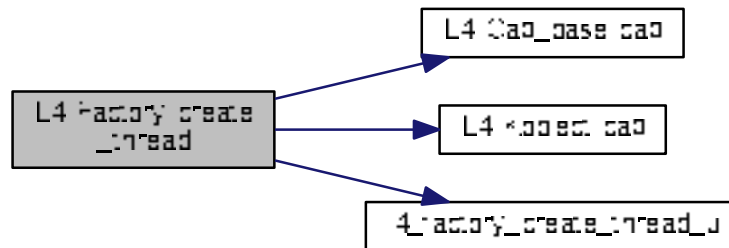
See also

[L4::Thread](#)

Definition at line 324 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_thread_u\(\)](#).

Here is the call graph for this function:



14.73.2.8 create_vm()

```
l4_msgtag_t L4::Factory::create_vm (
    Cap< Vm >const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

Create a new virtual machine.

Parameters

out	<i>target_cap</i>	The kernel stores the new VM's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

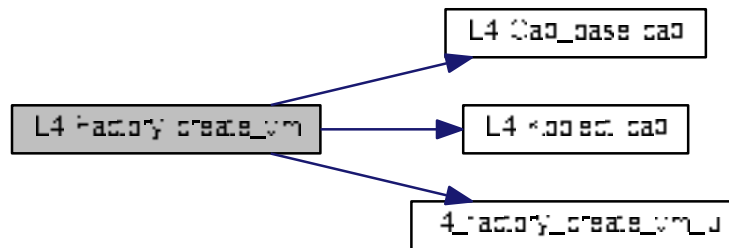
See also

[L4::Vm](#)

Definition at line 408 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_vm_u\(\)](#).

Here is the call graph for this function:



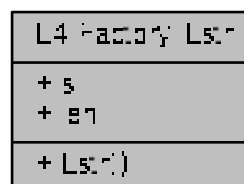
The documentation for this class was generated from the following file:

- [l4/sys/factory](#)

14.74 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

Collaboration diagram for L4::Factory::Lstr:



Public Member Functions

- [Lstr](#) (char const *[s](#), int [len](#))

Data Fields

- char const * [s](#)
The character buffer.
- int [len](#)
The number of characters in the buffer.

14.74.1 Detailed Description

Special type to add a pascal string into the factory create stream.

This encapsulates a string that has an explicit length.

Definition at line 71 of file [factory](#).

14.74.2 Constructor & Destructor Documentation

14.74.2.1 Lstr()

```
L4::Factory::Lstr::Lstr (
    char const * s,
    int len ) [inline]
```

Parameters

<i>s</i>	Pointer to the c-style string.
<i>len</i>	Length in number of characters of the string s.

Definition at line 87 of file [factory](#).

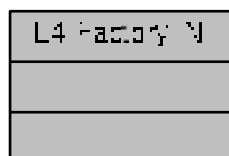
The documentation for this struct was generated from the following file:

- [l4/sys/factory](#)

14.75 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

Collaboration diagram for L4::Factory::Nil:



14.75.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 64 of file [factory](#).

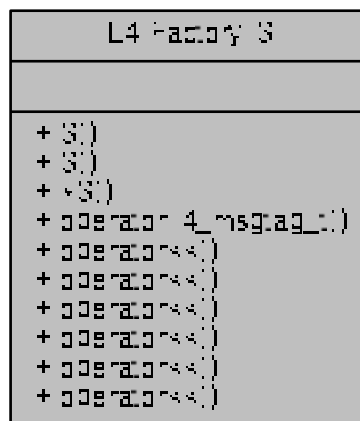
The documentation for this struct was generated from the following file:

- [l4/sys/factory](#)

14.76 L4::Factory::S Class Reference

Stream class for the [create\(\)](#) argument stream.

Collaboration diagram for L4::Factory::S:



Public Member Functions

- [S](#) ([S](#) const &o)
Create a copy.
- [S](#) ([l4_cap_idx_t](#) f, long obj, [L4::Cap](#)< void > target, [l4_utcb_t](#) *utcb) throw ()
Create a stream for a specific [create\(\)](#) call.
- [~S](#) ()
Commit the operation in the destructor to have a cool syntax for [create\(\)](#).
- [operator l4_msgtag_t](#) ()
Explicitly commits the operation and returns the result.
- [S](#) & [operator<<](#) ([l4_mword_t](#) i)
Put a single [l4_mword_t](#) as next argument.
- [S](#) & [operator<<](#) ([l4_umword_t](#) i)
Put a single [l4_umword_t](#) as next argument.

- [S](#) & [operator<<](#) (char const *s)
Add a zero-terminated string as next argument.
- [S](#) & [operator<<](#) ([Lstr](#) const &s)
Add a pascal string as next argument.
- [S](#) & [operator<<](#) ([Nil](#))
Add an empty argument.
- [S](#) & [operator<<](#) ([l4_fpage_t](#) d)
Add a flex page as next argument.

14.76.1 Detailed Description

Stream class for the [create\(\)](#) argument stream.

This stream allows a variable number of arguments to be added to a [create\(\)](#) call.

Definition at line 96 of file [factory](#).

14.76.2 Constructor & Destructor Documentation

14.76.2.1 [S\(\)](#) [1/2]

```
L4::Factory::S::S (
    S const & o ) [inline]
```

Create a copy.

Parameters

<i>o</i>	Instance of S to copy.
----------	--

Definition at line 109 of file [factory](#).

References [l4_msgtag_t::raw](#).

14.76.2.2 [S\(\)](#) [2/2]

```
L4::Factory::S::S (
    l4_cap_idx_t f,
    long obj,
    L4::Cap< void > target,
    l4_utcb_t * utcb ) throw () [inline]
```

Create a stream for a specific [create\(\)](#) call.

Parameters

	<i>f</i>	The capability for the factory object (L4::Factory).
	<i>obj</i>	The protocol ID to describe the type of the object that shall be created.
out	<i>target</i>	The capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new object's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Definition at line [124](#) of file [factory](#).

14.76.3 Member Function Documentation

14.76.3.1 operator l4_msgtag_t()

```
L4::Factory::S::operator l4_msgtag_t ( ) [inline]
```

Explicitly commits the operation and returns the result.

Returns

The result of the [create\(\)](#) operation.

Definition at line [144](#) of file [factory](#).

14.76.3.2 operator<<() [1/6]

```
S& L4::Factory::S::operator<< (
    l4_mword_t i ) [inline]
```

Put a single `l4_mword_t` as next argument.

Parameters

<i>i</i>	The value to add as next argument.
----------	------------------------------------

Returns

Reference to this stream.

Definition at line [158](#) of file [factory](#).

14.76.3.3 operator<<() [2/6]

```
S& L4::Factory::S::operator<< (
    l4_umword_t i ) [inline]
```

Put a single l4_umword_t as next argument.

Parameters

<i>i</i>	The value to add as next argument.
----------	------------------------------------

Returns

Reference to this stream.

Definition at line 171 of file [factory](#).

14.76.3.4 operator<<() [3/6]

```
S& L4::Factory::S::operator<< (
    char const * s ) [inline]
```

Add a zero-terminated string as next argument.

Parameters

<i>s</i>	The string to add as next argument.
----------	-------------------------------------

Returns

Reference to this stream.

The string will be added with the zero-terminator.

Definition at line 186 of file [factory](#).

14.76.3.5 operator<<() [4/6]

```
S& L4::Factory::S::operator<< (
    Lstr const & s ) [inline]
```

Add a pascal string as next argument.

Parameters

<code>s</code>	The string to add as next argument.
----------------	-------------------------------------

Returns

Reference to this stream.

The string will be added with the exact length given. It is the responsibility of the caller to make sure that the string is zero-terminated when that is required by the server.

Definition at line 203 of file [factory](#).

14.76.3.6 operator<<() [5/6]

```
S& L4::Factory::S::operator<< (
    Nil ) [inline]
```

Add an empty argument.

Returns

Reference to this stream.

Definition at line 214 of file [factory](#).

14.76.3.7 operator<<() [6/6]

```
S& L4::Factory::S::operator<< (
    l4_fpage_t d ) [inline]
```

Add a flex page as next argument.

Parameters

<code>d</code>	The flex page to add (there will be no map operation).
----------------	--

Returns

Reference to this stream.

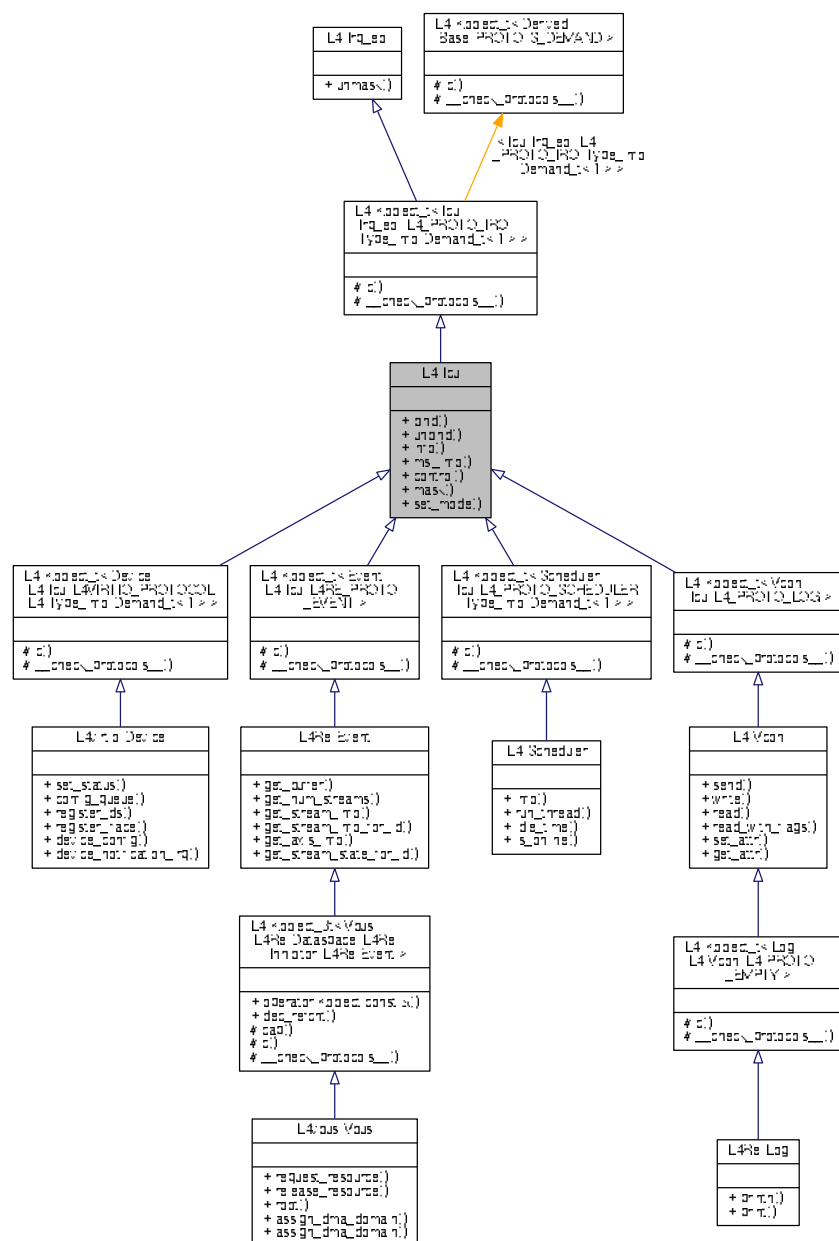
Definition at line 227 of file [factory](#).

The documentation for this class was generated from the following file:

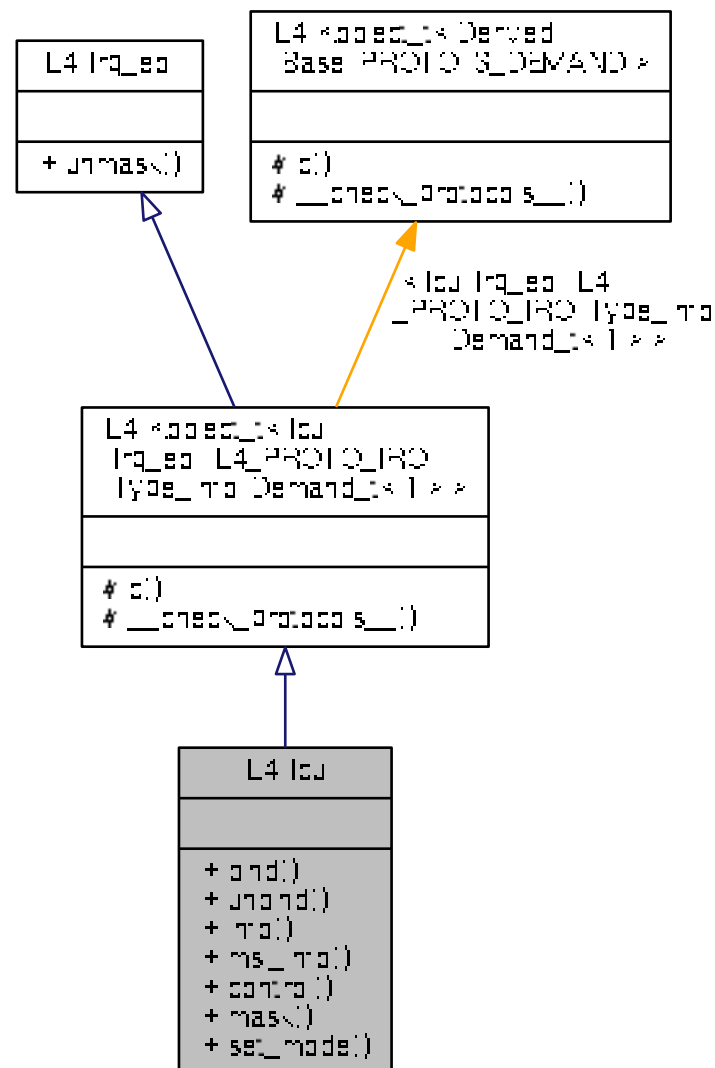
- [l4/sys/factory](#)

C++ **lcu** interface.

Inheritance diagram for L4::lcu:



Collaboration diagram for L4::Icu:



Data Structures

- class [Info](#)

This class encapsulates information about an ICU.

Public Member Functions

- [I4_msgtag_t bind](#) (unsigned irqnum, [L4::Cap< Triggerable > irq](#), [I4_utcb_t *utcb=I4_utcb\(\)](#)) throw ()
Bind an interrupt line of an interrupt controller to an interrupt object.
- [I4_msgtag_t unbind](#) (unsigned irqnum, [L4::Cap< Triggerable > irq](#), [I4_utcb_t *utcb=I4_utcb\(\)](#)) throw ()
Remove binding of an interrupt line from the interrupt controller object.

- [l4_msgtag_t info](#) ([l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Get information about the capabilities of the ICU.
- [l4_msgtag_t msi_info](#) ([l4_umword_t](#) irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info)
Get MSI info about IRQ.
- [l4_msgtag_t mask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Mask an IRQ line.
- [l4_msgtag_t set_mode](#) (unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Set interrupt mode.

Additional Inherited Members

14.77.1 Detailed Description

C++ [Icu](#) interface.

Note

"ICU" is short for "interrupt control unit".

This class defines the interface for interrupt controllers. It defines functions for binding [L4::Irq](#) objects to interrupt lines, as well as functions for masking and unmasking of interrupts.

To setup an interrupt line the following steps are required:

1. [set_mode\(\)](#) (optional if interrupt has a default mode)
2. [Irq::attach\(\)](#) to attach the interrupt capability to a thread
3. [bind\(\)](#)
4. [unmask\(\)](#) to receive the first interrupt

Include File

```
#include <l4/sys/icu>
```

Definition at line 262 of file [irq](#).

14.77.2 Member Function Documentation

14.77.2.1 bind()

```
l4\_msgtag\_t L4::Icu::bind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4\_utcb\_t * utcb = l4\_utcb\(\) ) throw ()    [inline]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object for the given IRQ line to bind to this ICU.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

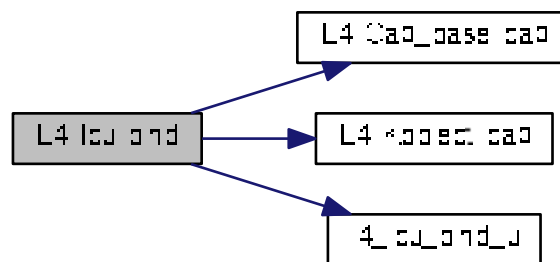
Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::icu::unmask](#).

Definition at line 310 of file [irq](#).

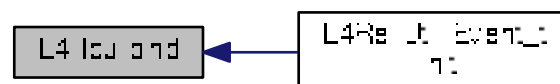
References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [l4_icu_bind_u\(\)](#), [L4_ICU_OP_BIND](#), and [L4_RPC_NF_OP](#).

Referenced by [L4Re::Util::Event_t< PAYLOAD >::init\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.77.2.2 info()

```
l4_msgtag_t L4::Icu::info (
    l4_icu_info_t * info,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get information about the capabilities of the ICU.

Parameters

<i>out</i>	<i>info</i>	Info structure to be filled with information.
	<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

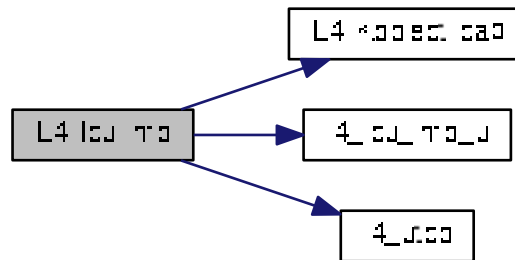
Returns

Syscall return tag

Definition at line 341 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [l4_icu_info_u\(\)](#), [L4_ICU_OP_INFO](#), [L4_ICU_OP_MSI_INFO](#), [L4_INLINE_RPC_OP](#), [L4_RPC_NF_OP](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



14.77.2.3 mask()

```

l4_msgtag_t L4::Icu::mask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Mask an IRQ line.

Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If NULL this function is a send-only message to the ICU. If not NULL this function will enter an open wait after sending the mask message.
<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

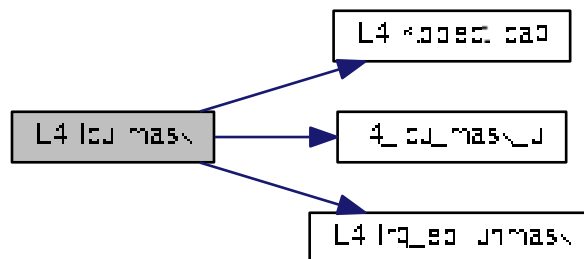
Returns

Syscall return tag

Definition at line 384 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [l4_icu_mask_u\(\)](#), [L4_ICU_OP_MASK](#), [L4_ICU_OP_UNMASK](#), [L4_RPC_NF_OP](#), and [L4::Irq_eoi::unmask\(\)](#).

Here is the call graph for this function:



14.77.2.4 msi_info()

```

l4_msgtag_t L4::Icu::msi_info (
    l4_umword_t irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info )
  
```

Get MSI info about IRQ.

Parameters

	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

Returns

Syscall return tag

14.77.2.5 `set_mode()`

```

14_msgtag_t L4::Icu::set_mode (
    unsigned irqnum,
    14_umword_t mode,
    14_utcb_t * utcb = 14_utcb() ) throw ()    [inline]

```

Set interrupt mode.

Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see L4_irq_mode .
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

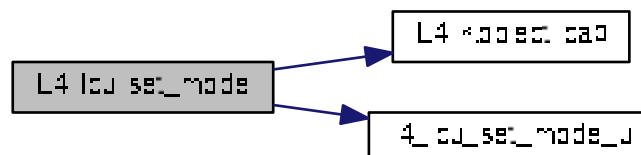
Returns

Syscall return tag

Definition at line 406 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [L4_ICU_OP_SET_MODE](#), [l4_icu_set_mode_u\(\)](#), and [L4_RPC_NF_OP](#).

Here is the call graph for this function:

14.77.2.6 `unbind()`

```

14_msgtag_t L4::Icu::unbind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    14_utcb_t * utcb = 14_utcb() ) throw ()    [inline]

```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

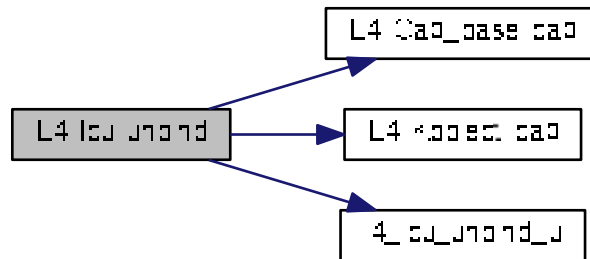
Returns

Syscall return tag

Definition at line 326 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [L4_ICU_OP_UNBIND](#), [I4_icu_unbind_u\(\)](#), and [L4_RPC_NF_OP](#).

Here is the call graph for this function:



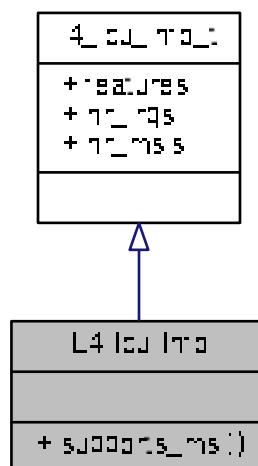
The documentation for this class was generated from the following file:

- [I4/sys/irq](#)

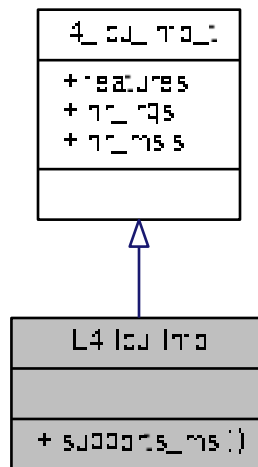
14.78 L4::Icu::Info Class Reference

This class encapsulates information about an ICU.

Inheritance diagram for L4::Icu::Info:



Collaboration diagram for L4::Icu::Info:



Additional Inherited Members

14.78.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line 289 of file [irq](#).

The documentation for this class was generated from the following file:

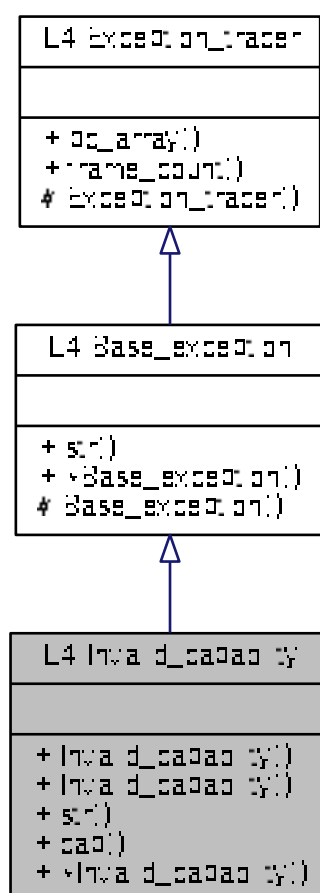
- [l4/sys/irq](#)

14.79 L4::Invalid_capability Class Reference

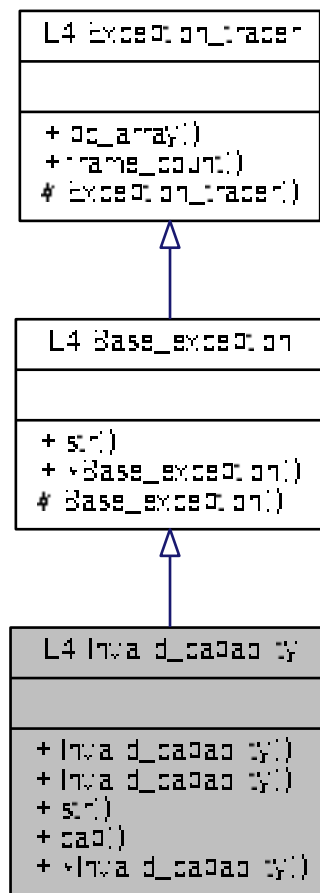
Indicates that an invalid object was invoked.

```
#include <l4/cxx/exceptions>
```


Inheritance diagram for L4::Invalid_capability:



Collaboration diagram for L4::Invalid_capability:



Public Member Functions

- **Invalid_capability** (Cap< void > const &o) throw ()
Create an Invalid_obejct exception for the Object o.
- char const * **str** () const throw ()
Return a human readable string for the exception.
- Cap< void > const & **cap** () const throw ()
Get the object that caused the error.

Additional Inherited Members

14.79.1 Detailed Description

Indicates that an invalid object was invoked.

An Object is invalid if it has L4_INVALID_ID as server [L4](#) UID, or if the server does not know the object ID.

Definition at line [245](#) of file [exceptions](#).

14.79.2 Constructor & Destructor Documentation

14.79.2.1 Invalid_capability()

```
L4::Invalid_capability::Invalid_capability (
    Cap< void > const & o ) throw ()    [inline], [explicit]
```

Create an Invalid_obejct exception for the Object o.

Parameters

<i>o</i>	The object that caused the server side error.
----------	---

Definition at line 255 of file [exceptions](#).

References [L4::Cap_base::cap\(\)](#).

Here is the call graph for this function:



14.79.3 Member Function Documentation

14.79.3.1 cap()

```
Cap<void> const& L4::Invalid_capability::cap ( ) const throw ()    [inline]
```

Get the object that caused the error.

Returns

The object that caused the error on invocation.

Definition at line 264 of file [exceptions](#).

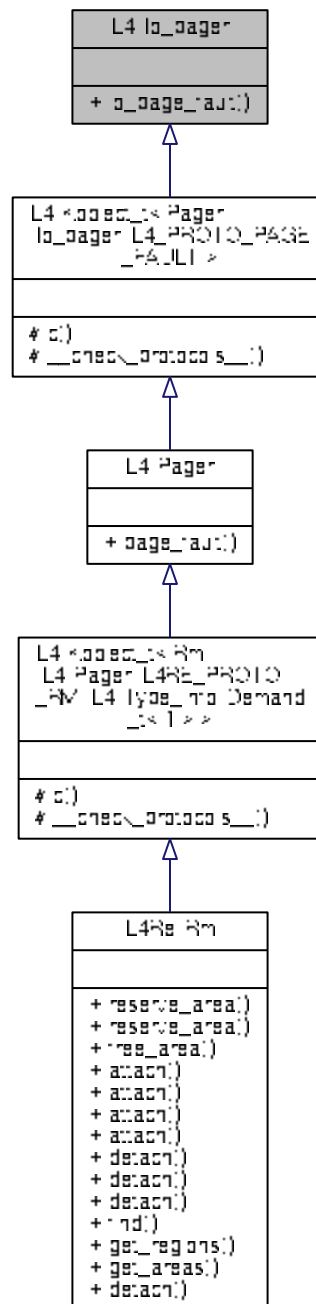
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

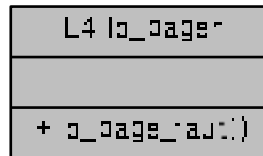
14.80 L4::lo_pager Class Reference

[lo_pager](#) interface.

Inheritance diagram for L4::lo_pager:



Collaboration diagram for L4::io_pager:



Public Member Functions

- [l4_msgtag_t io_page_fault](#) ([l4_fpage_t](#) io_pfa, [l4_umword_t](#) pc, [L4::lpc::Opt](#)< [l4_mword_t](#) &> result, [L4::lpc::Rcv_fpage](#) rwin, [L4::lpc::Opt](#)< [L4::lpc::Snd_fpage](#) &> fp)
IO page fault protocol message.

14.80.1 Detailed Description

[io_pager](#) interface.

Note

This interface is IA32 specific.

This class defines the interface for handling IO page faults. IO page faults happen when a thread tries to access an IO port that it does not currently have access to.

Depending on the microkernel's implementation, IO page faults can be handled in two ways.

If the microkernel does not support IO page faults, this IO pagefault interface is not used. Instead, the microkernel sends an exception IPC to the thread's exception handler ([L4::Exception](#)), indicating a #GP (exception number 13). The exception handler must consult the faulting instruction to determine the cause of the exception. This is the default in Fiasco.OC.

In contrast, if the microkernel supports IO page faults, the microkernel will generate an IO page fault message and send it to the thread's page fault handler (pager). The page fault handler can implement this interface to handle the IO page faults.

Note

A program may use this mechanism to implement a lazy IO port access scheme.
The page fault and exception handlers are set with the [L4::Thread::control](#) interface.

Definition at line 61 of file [pager](#).

14.80.2 Member Function Documentation

14.80.2.1 `io_page_fault()`

```
l4_msgtag_t L4::Io_pager::io_page_fault (
    l4_fpage_t io_pfa,
    l4_umword_t pc,
    L4::Ipc::Opt< l4_mword_t > result,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage > fp )
```

IO page fault protocol message.

Parameters

	<i>io_pfa</i>	Flex-page describing the faulting IO-port.
	<i>pc</i>	Faulting program counter.
out	<i>result</i>	Optional: handling result value.
	<i>rwin</i>	The receive window for a flex-page mapping.
out	<i>fp</i>	Optional: flex-page descriptor to send to the task raising the page fault.

Returns

System call message tag; use [l4_error\(\)](#) to check for errors.

IO-port fault messages are usually generated by the kernel and an IO-page-fault handler needs to be in place to handle such faults and generate a reply by filling in `result` and / or `fp`.

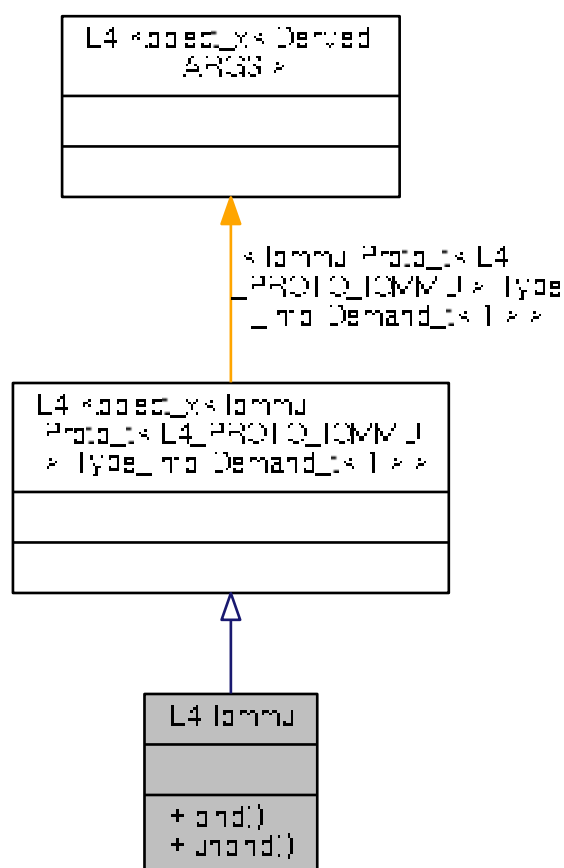
The documentation for this class was generated from the following file:

- [l4/sys/pager](#)

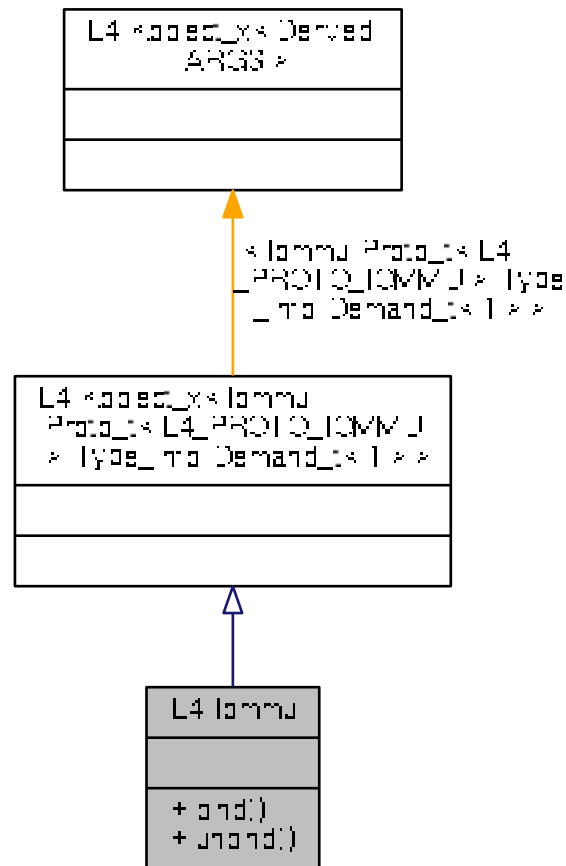
14.81 L4::lommu Class Reference

Interface for IO-MMUs used for DMA remapping.

Inheritance diagram for L4::lomu:



Collaboration diagram for L4::Iommu:



Public Member Functions

- `I4_msgtag_t bind (I4_uint64_t src_id, lpc::Cap< Task > dma_space)`
Associate `dma_space` with the set of device(s) specified by `src_id`.
- `I4_msgtag_t unbind (I4_uint64_t src_id, lpc::Cap< Task > dma_space)`
Remove the association of the given DMA address space from the device(s) specified by `src_id`.

14.81.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

This interface allows to associate a DMA address space with a platform dependent set of devices.

Definition at line 16 of file `iommu`.

14.81.2 Member Function Documentation

14.81.2.1 bind()

```
l4_msgtag_t L4::Iommu::bind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space )
```

Associate `dma_space` with the set of device(s) specified by `src_id`.

Parameters

<i>src_id</i>	Platform dependent source ID specifying the set of devices that shall use <code>dma_space</code> for DMA remapping.
<i>dma_space</i>	The DMA space (L4::Task created with <code>L4_PROTO_DMA_SPACE</code>) providing the mappings that shall be used for the device(s).

14.81.2.2 unbind()

```
l4_msgtag_t L4::Iommu::unbind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space )
```

Remove the association of the given DMA address space from the device(s) specified by `src_id`.

Parameters

<i>src_id</i>	Platform dependent source ID specifying the set of devices that shall no longer use <code>dma_space</code> for DMA remapping.
<i>dma_space</i>	The DMA space formerly associated with bind() .

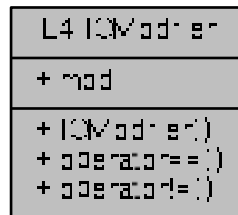
The documentation for this class was generated from the following file:

- `l4/sys/iommu`

14.82 L4::IOModifier Class Reference

Modifier class for the IO stream.

Collaboration diagram for L4::IOModifier:



14.82.1 Detailed Description

Modifier class for the IO stream.

An IO Modifier can be used to change properties of an IO stream for example the number format.

Definition at line 33 of file [basic_ostream](#).

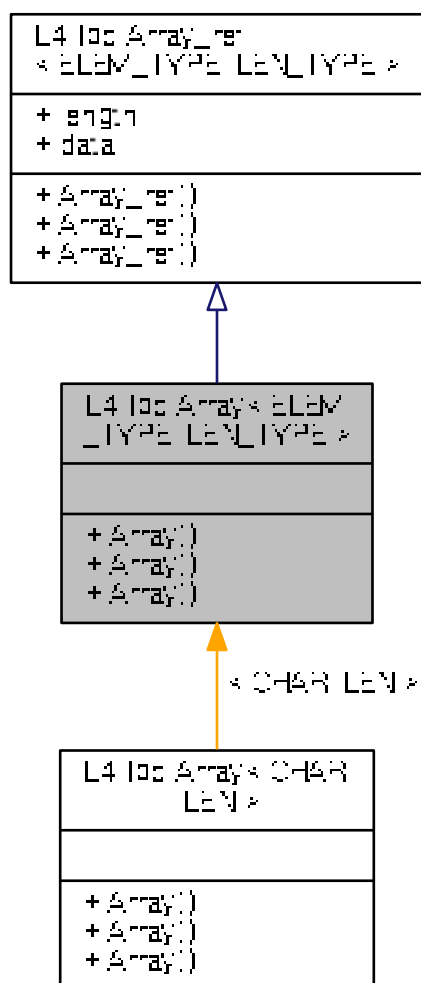
The documentation for this class was generated from the following file:

- [l4/cxx/basic_ostream](#)

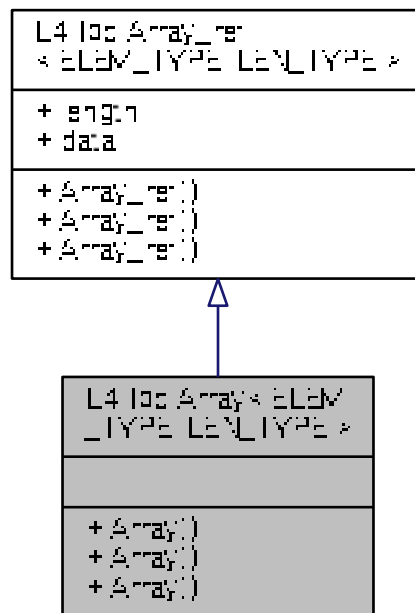
14.83 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference

[Array](#) data type for dynamically sized arrays in RPCs.

Inheritance diagram for L4::lpc::Array< ELEM_TYPE, LEN_TYPE >:



Collaboration diagram for L4::ipc::Array< ELEM_TYPE, LEN_TYPE >:



Public Member Functions

- [Array](#) ()
Make array.
- [Array](#) (LEN_TYPE length, ELEM_TYPE *data)
Make array from length and data pointer.
- [Array](#) (typename Non_const< ELEM_TYPE >::type const &other)
Make a const array from a non-const array.

14.83.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::ipc::Array< ELEM_TYPE, LEN_TYPE >
```

[Array](#) data type for dynamically sized arrays in RPCs.

Template Parameters

<i>ELEM_TYPE</i>	The data type of an array element, should be 'const' when used as input.
<i>LEN_TYPE</i>	Data type used to store the number of elements in the array.

An [Array](#) generally encapsulates a data pointer and a length (number of elements). [Array](#) does *not* provide any

storage for the data itself. The storage is either provided by a client-side caller or in the case of [Array_ref](#) is the message itself.

Arrays can be used as input or as output arguments, when used as input `ELEM_TYPE` should be qualified *const*, when used as output a reference to an array must be used and the `ELEM_TYPE` must *not* be qualified *const*. It is the caller's responsibility to provide an array buffer of sufficient length. If a message from the server is too large it will be silently truncated.

If backward compatibility with `lpc::Stream` is required, then `LEN_TYPE` must be `unsigned long`.

Definition at line 85 of file [ipc_array](#).

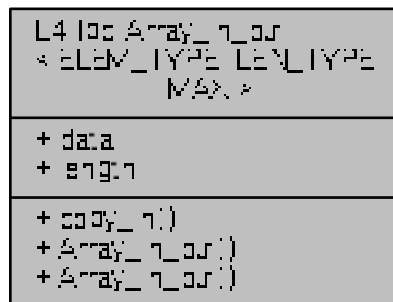
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

14.84 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference

Server-side copy in buffer for [Array](#).

Collaboration diagram for L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >:



Public Member Functions

- `void copy_in (const_array a)`
copy in data from a source array
- `Array_in_buf (const_array a)`
Make [Array_in_buf](#) from a const array.
- `Array_in_buf (array a)`
Make [Array_in_buf](#) from a non-const array.

Data Fields

- ELEM_TYPE [data](#) [MAX]
The data elements.
- LEN_TYPE [length](#)
The length of the array.

14.84.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default, LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t)) / sizeof(ELEM_TYPE)>
struct L4::ipc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >
```

Server-side copy in buffer for [Array](#).

Template Parameters

<i>ELEM_TYPE</i>	Data type of an array element.
<i>LEN_TYPE</i>	Data type for the number of elements in the array.
<i>MAX</i>	The maximum number of elements in the buffer. If the actual message is longer than the buffer, it will be silently truncated.

This type is assignment compatible to `Array_ref<ELEM_TYPE, LEN_TYPE>` and provides a transparent server-side copy-in mechanism for array parameters. The [Array_in_buf](#) provides the storage for the array data and receives a copy of the data passed to the server-function.

Definition at line [123](#) of file [ipc_array](#).

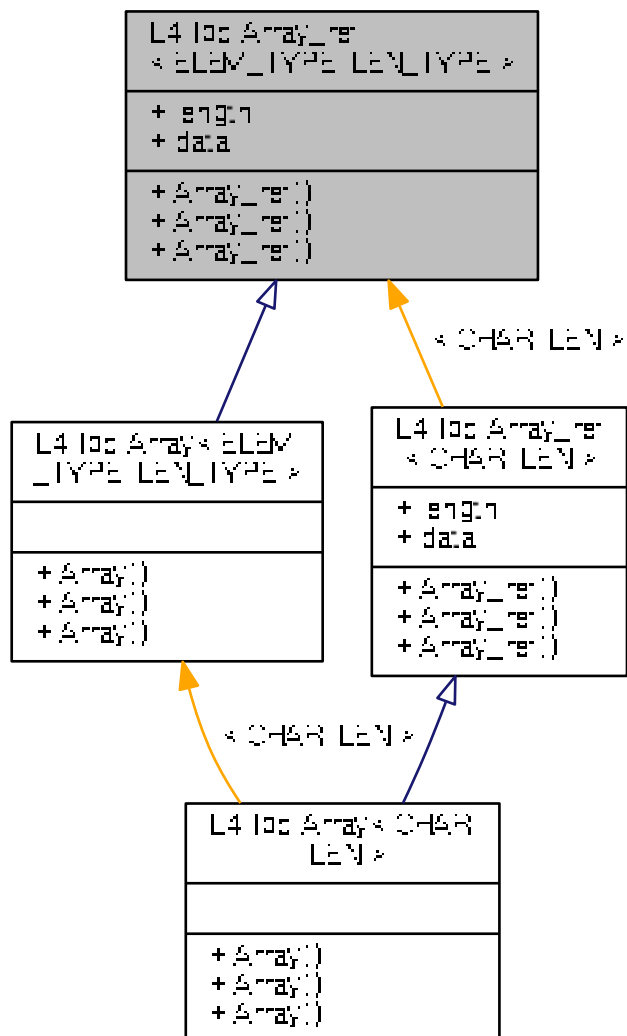
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

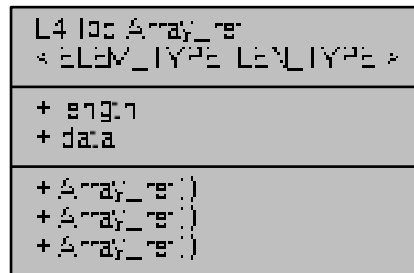
14.85 L4::ipc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference

[Array](#) reference data type for arrays located in the message.

Inheritance diagram for L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >:



Collaboration diagram for L4::ipc::Array_ref< ELEM_TYPE, LEN_TYPE >:



14.85.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::ipc::Array_ref< ELEM_TYPE, LEN_TYPE >
```

[Array](#) reference data type for arrays located in the message.

Note

Use [Array](#) for normal RPC interfaces, [Array_ref](#) is usually used as server-side argument, see [Array](#).

Template Parameters

<i>ELEM_TYPE</i>	The data type of an array element, should be 'const' when used as input.
<i>LEN_TYPE</i>	Data type used to store the number of elements in the array.

Definition at line 39 of file [ipc_array](#).

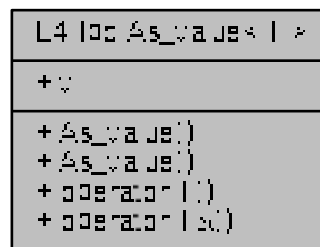
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

14.86 L4::ipc::As_value< T > Struct Template Reference

Pass the argument as plain data value.

Collaboration diagram for L4::lpc::As_value< T >:



14.86.1 Detailed Description

```
template<typename T>
struct L4::lpc::As_value< T >
```

Pass the argument as plain data value.

Template Parameters

<i>T</i>	The type of the original argument.
----------	------------------------------------

`As_value<T>` is used when *T* would be otherwise interpreted specially, for example as flex page. When using `As_value<>` then the argument is transmitted as plain data element.

Definition at line 127 of file [ipc_types](#).

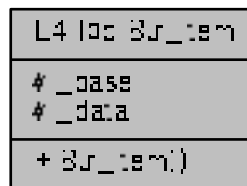
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.87 L4::lpc::Buf_item Class Reference

RPC warpper for a receive item.

Collaboration diagram for L4::ipc::Buf_item:



14.87.1 Detailed Description

RPC warpper for a receive item.

Definition at line 305 of file [ipc_types](#).

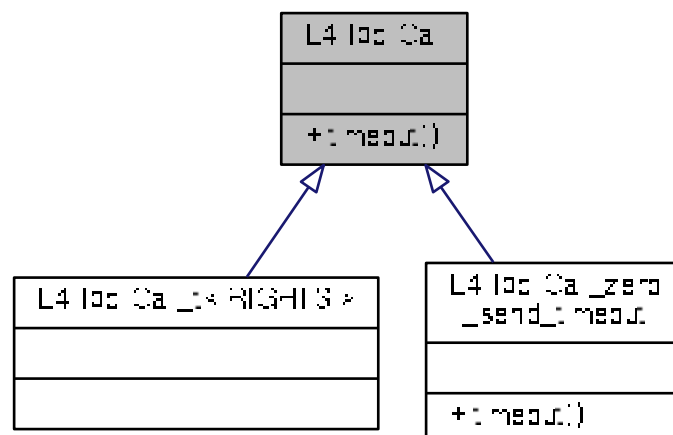
The documentation for this class was generated from the following file:

- [I4/sys/cxx/ipc_types](#)

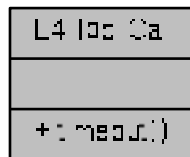
14.88 L4::ipc::Call Struct Reference

RPC attribute for a standard RPC call.

Inheritance diagram for L4::ipc::Call:



Collaboration diagram for L4::Ipc::Call:



14.88.1 Detailed Description

RPC attribute for a standard RPC call.

This is the default for the *FLAGS* parameter for L4::Ipc::Msg::Rpc_call L4::Ipc::Msg::Rpc_inline_call templates and declares the RPC to have default call semantics and timeouts.

Examples:

```
L4_RPC(long, send, (unsigned value), L4::Ipc::Call);
```

which is equivalent to:

```
L4_RPC(long, send, (unsigned value));
```

Definition at line 215 of file [ipc_iface](#).

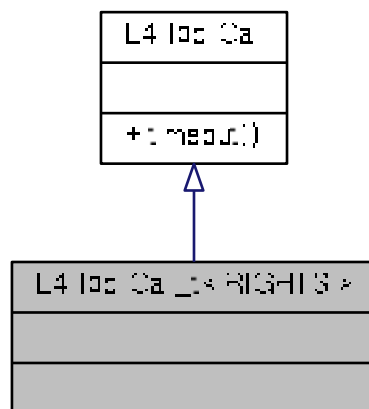
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

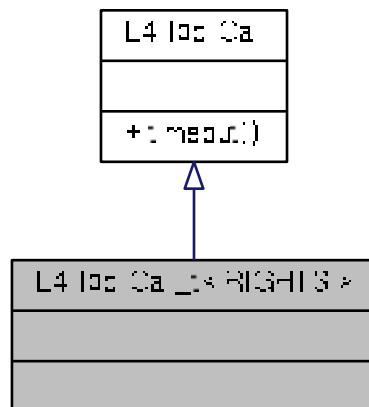
14.89 L4::Ipc::Call_t< RIGHTS > Struct Template Reference

RPC attribute for an RPC call with required rights.

Inheritance diagram for L4::ipc::Call_t< RIGHTS >:



Collaboration diagram for L4::ipc::Call_t< RIGHTS >:



14.89.1 Detailed Description

```
template<unsigned RIGHTS>
struct L4::ipc::Call_t< RIGHTS >
```

RPC attribute for an RPC call with required rights.

Template Parameters

<i>RIGHTS</i>	The capability rights required for this call. L4_CAP_FPAGE_W and L4_CAP_FPAGE_S are checked within the server (and -L4_EPERM shall be returned if the caller has insufficient rights). L4_CAP_FPAGE_R is always on but might be specified for documentation purposes. Other rights cannot be used in this context, because they cannot be checked at the server side.
---------------	---

Examples:

```
L4_RPC(long, func, (unsigned value), L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
```

Definition at line 246 of file [ipc_iface](#).

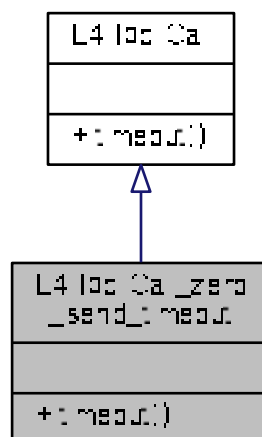
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

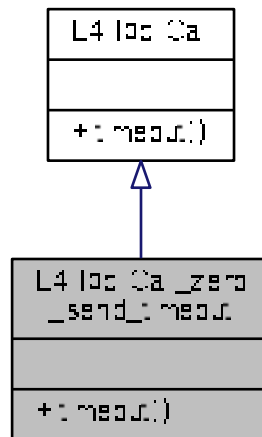
14.90 L4::lpc::Call_zero_send_timeout Struct Reference

RPC attribute for an RPC call, with zero send timeout.

Inheritance diagram for L4::lpc::Call_zero_send_timeout:



Collaboration diagram for `L4::ipc::Call_zero_send_timeout`:



14.90.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

Definition at line 225 of file [ipc_iface](#).

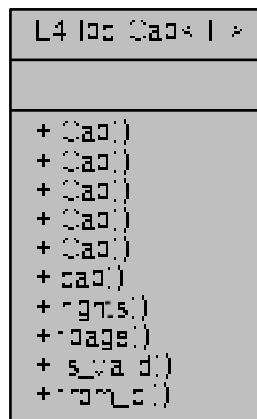
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

14.91 `L4::ipc::Cap< T >` Class Template Reference

Capability type for RPC interfaces (see [L4::Cap<T>](#)).

Collaboration diagram for L4::lpc::Cap< T >:



Public Types

- enum { [Rights_mask](#) = 0xff, [Cap_mask](#) = L4_CAP_MASK }

Public Member Functions

- template<typename O >
[Cap](#) ([Cap](#)< O > const &o)
Make copy with conversion.
- [Cap](#) (L4::Cap< T > cap)
Make a [Cap](#) from L4::Cap< T >, with minimal rights.
- template<typename O >
[Cap](#) (L4::Cap< O > cap)
Make IPC [Cap](#) from L4::Cap with conversion (and minimal rights).
- [Cap](#) ()
Make an invalid cap.
- [Cap](#) (L4::Cap< T > cap, unsigned char rights)
Make a [Cap](#) from L4::Cap< T > with the given rights.
- L4::Cap< T > cap () const
Return the L4::Cap< T > of this [Cap](#).
- unsigned rights () const
Return the rights bits stored in this IPC cap.
- L4::lpc::Snd_fpage fpage () const
Return the send flexpage for this [Cap](#) (see [l4_fpage_t](#))
- bool is_valid () const throw ()
Return true if this [Cap](#) is valid.

Static Public Member Functions

- static [Cap from_ci](#) ([l4_cap_idx_t](#) c)
Create an IPC capability from a C capability index plus rights.

14.91.1 Detailed Description

```
template<typename T>
class L4::lpc::Cap< T >
```

Capability type for RPC interfaces (see [L4::Cap<T>](#)).

Template Parameters

T	type of the interface referenced by the capability.
-------------------	---

In contrast to [L4::Cap<T>](#) this type additionally stores a rights mask that shall be used when the capability is transferred to the receiver. This allows to apply restrictions to the transferred capability in the form of a subset of the rights possessed by the sender.

See also

[L4::lpc::make_cap\(\)](#)

Definition at line [541](#) of file [ipc_types](#).

14.91.2 Member Enumeration Documentation

14.91.2.1 anonymous enum

```
template<typename T>
anonymous enum
```

Enumerator

Rights_mask	Mask for rights bits stored internally. L4_FPAGE_RIGHTS_MASK L4_FPAGE_C_NO_REF_CNT L4_FPAGE_C_OBJ_RIGHTS).
Cap_mask	Mask for significant capability bits. (incl. the invalid bit to support invalid caps)

Definition at line [547](#) of file [ipc_types](#).

14.91.3 Constructor & Destructor Documentation

14.91.3.1 Cap()

```
template<typename T>
L4::lpc::Cap< T >::Cap (
    L4::lpc::Cap< T > cap,
    unsigned char rights ) [inline]
```

Make a [Cap](#) from `L4::Cap<T>` with the given rights.

Parameters

<i>cap</i>	Capability to be sent.
<i>rights</i>	Rights to be sent. Consists of L4_fpage_rights and L4_obj_fpage_ctl .

Definition at line 589 of file [ipc_types](#).

14.91.4 Member Function Documentation

14.91.4.1 from_ci()

```
template<typename T>
static Cap L4::lpc::Cap< T >::from_ci (
    l4_cap_idx_t c ) [inline], [static]
```

Create an IPC capability from a C capability index plus rights.

Parameters

<i>c</i>	C capability index with the lowest 8 bits used as rights for the map operation (see L4_fpage_rights).
----------	--

Definition at line 597 of file [ipc_types](#).

The documentation for this class was generated from the following file:

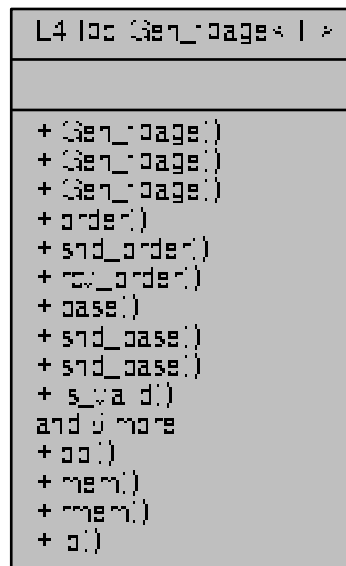
- [l4/sys/cxx/ipc_types](#)

14.92 L4::lpc::Gen_fpage< T > Class Template Reference

Generic RPC wrapper for [L4](#) flex-pages.

Inherits `T`.

Collaboration diagram for L4::lpc::Gen_fpage< T >:



Public Types

- enum [Type](#)
Type of mapping object, see `L4_fpage_type`.
- enum [Map_type](#)
Kind of mapping.
- enum [Cacheopt](#)
Caching options, see `L4_fpage_cacheability_opt_t`.

Public Member Functions

- bool [is_valid](#) () const
Check if the capability is valid.
- bool [cap_received](#) () const
Check if the capability has been mapped.
- bool [id_received](#) () const
Check if a label was received instead of a mapping.
- bool [local_id_received](#) () const
Check if a local capability id has been received.
- bool [is_compound](#) () const
Check if the received item has the compound bit set.
- [l4_umword_t data](#) () const
Return the raw flex page descriptor.
- [l4_umword_t base_x](#) () const
Return the raw base descriptor.

14.92.1 Detailed Description

```
template<typename T>
class L4::lpc::Gen_fpage< T >
```

Generic RPC wrapper for [L4](#) flex-pages.

Template Parameters

<i>T</i>	Underlying specific flexpage type.
----------	------------------------------------

Definition at line [321](#) of file [ipc_types](#).

14.92.2 Member Function Documentation

14.92.2.1 cap_received()

```
template<typename T>
bool L4::lpc::Gen_fpage< T >::cap_received ( ) const [inline]
```

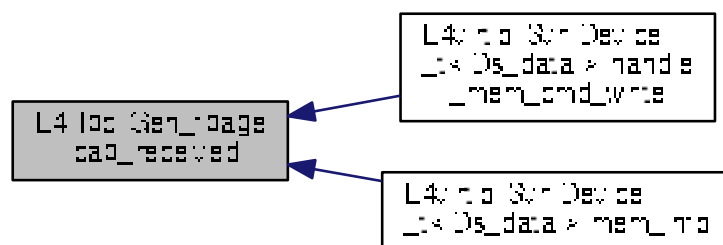
Check if the capability has been mapped.

The capability itself can then be retrieved from the cap slot that has been provided in the receive operation.

Definition at line [438](#) of file [ipc_types](#).

Referenced by [L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write\(\)](#), and [L4virtio::Svr::Device_t< Ds_data >::mem_info\(\)](#).

Here is the caller graph for this function:



14.92.2.2 id_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::id_received ( ) const [inline]
```

Check if a label was received instead of a mapping.

For IPC gates, if the L4_RCV_ITEM_LOCAL_ID has been set, then only the label of the IPC gate will be provided if the gate is local to the receiver, i.e. the target thread of the IPC gate is in the same task as the receiving thread.

The label can be retrieved with [Gen_fpage::data\(\)](#).

Definition at line 450 of file [ipc_types](#).

Referenced by [L4Re::Util::Dataspace_svr::is_static\(\)](#).

Here is the caller graph for this function:



14.92.2.3 is_compound()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::is_compound ( ) const [inline]
```

Check if the received item has the compound bit set.

A set compound bit means the next message item of the same type will be mapped to the same receive buffer as this message item.

Definition at line 468 of file [ipc_types](#).

14.92.2.4 local_id_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::local_id_received ( ) const [inline]
```

Check if a local capability id has been received.

If the L4_RCV_ITEM_LOCAL_ID flag has been set by the receiver, and sender and receiver are in the same task, then only the capability index is transferred.

The capability can be retrieved with [Gen_fpage::data\(\)](#).

Definition at line 460 of file [ipc_types](#).

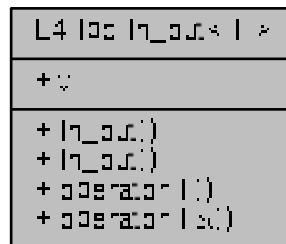
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.93 L4::lpc::In_out< T > Struct Template Reference

Mark an argument as in-out argument.

Collaboration diagram for L4::lpc::In_out< T >:



14.93.1 Detailed Description

```
template<typename T>
struct L4::lpc::In_out< T >
```

Mark an argument as in-out argument.

Template Parameters

<i>T</i>	The original argument type, usually a pointer or a reference.
----------	---

In_out<> is used when an otherwise output-only value shall also be used as input value.

Definition at line 52 of file [ipc_types](#).

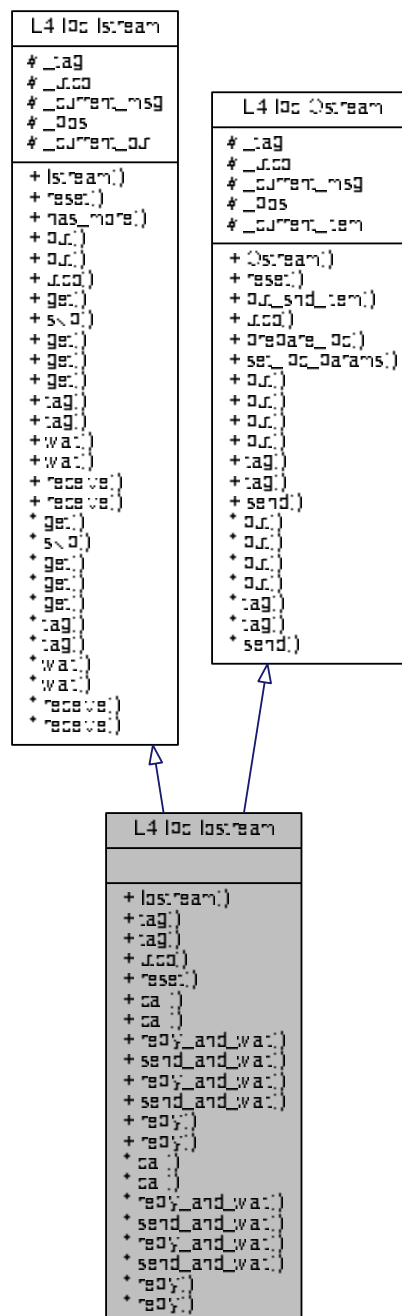
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

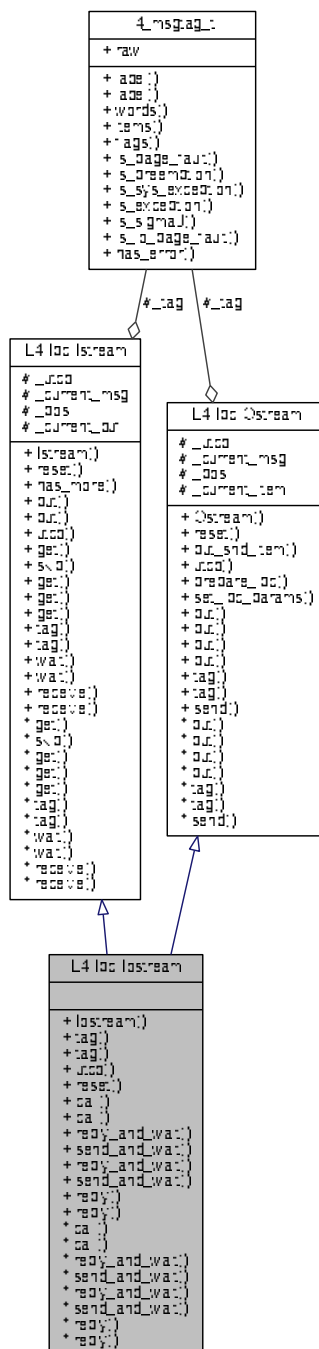
14.94 L4::lpc::lostream Class Reference

Input/Output stream for IPC [un]marshalling.

Inheritance diagram for L4::lpc::lostream:



Collaboration diagram for L4::lpc::loststream:



Public Member Functions

- `lostream (l4_utcb_t *utcb)`
Create an IPC IO stream with a single message buffer.
- `void reset ()`
Reset the stream to its initial state.

IPC operations.

- [l4_msgtag_t call](#) ([l4_cap_idx_t](#) dst, [l4_timeout_t](#) timeout, long proto=0)
Do an IPC call using the message in the output stream and receiving to the input stream.
- [l4_msgtag_t call](#) ([l4_cap_idx_t](#) dst, long proto=0)
- [l4_msgtag_t reply_and_wait](#) ([l4_umword_t](#) *src_dst, long proto=0)
Do an IPC reply and wait.
- [l4_msgtag_t send_and_wait](#) ([l4_cap_idx_t](#) dest, [l4_umword_t](#) *src, long proto=0)
- [l4_msgtag_t reply_and_wait](#) ([l4_umword_t](#) *src_dst, [l4_timeout_t](#) timeout, long proto=0)
Do an IPC reply and wait.
- [l4_msgtag_t send_and_wait](#) ([l4_cap_idx_t](#) dest, [l4_umword_t](#) *src, [l4_timeout_t](#) timeout, long proto=0)
- [l4_msgtag_t reply](#) ([l4_timeout_t](#) timeout, long proto=0)
- [l4_msgtag_t reply](#) (long proto=0)

14.94.1 Detailed Description

Input/Output stream for IPC [un]marshalling.

The [lpc::lostream](#) is part of the AW Env IPC framework as well as [lpc::lstream](#) and [lpc::Ostream](#). In particular an [lpc::lostream](#) is a combination of an [lpc::lstream](#) and an [lpc::Ostream](#). It can use either a single message buffer for receiving and sending messages or a pair of a receive and a send buffer. The stream also supports combined IPC operations such as [call\(\)](#) and [reply_and_wait\(\)](#), which can be used to implement RPC functionality.

Examples:

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 793 of file [ipc_stream](#).

14.94.2 Constructor & Destructor Documentation**14.94.2.1 lostream()**

```
L4::Ipc::Iostream::Iostream (
    l4_utcb_t * utcb ) [inline], [explicit]
```

Create an IPC IO stream with a single message buffer.

Parameters

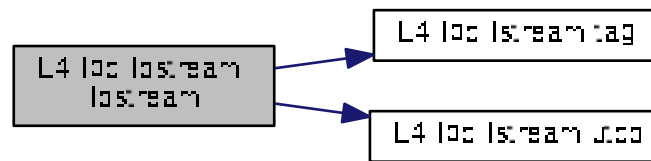
utcb	The message buffer used as backing store.
----------------------	---

The created IO stream uses the same message buffer for sending and receiving IPC messages.

Definition at line 805 of file [ipc_stream](#).

References [L4::lpc::lstream::tag\(\)](#), and [L4::lpc::lstream::utcb\(\)](#).

Here is the call graph for this function:



14.94.3 Member Function Documentation

14.94.3.1 `call()`

```
l4_msgtag_t L4::Ipc::Iostream::call (
    l4_cap_idx_t dst,
    l4_timeout_t timeout,
    long proto = 0 ) [inline]
```

Do an IPC call using the message in the output stream and receiving to the input stream.

Parameters

<i>dst</i>	The destination L4 UID (thread) to call.
<i>timeout</i>	The IPC timeout for the call.
<i>proto</i>	The protocol value to use in the message tag.

Returns

The result dope of the IPC operation.

This is a combined IPC operation consisting of a send and a receive to/from the given destination `dst`.

A call is usually used by clients for RPCs to a server.

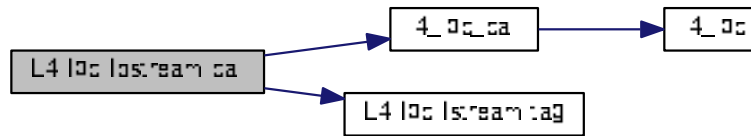
Examples:

[examples/libs/l4re/streammap/client.cc](#).

Definition at line [964](#) of file [ipc_stream](#).

References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), and [L4::lpc::Iostream::tag\(\)](#).

Here is the call graph for this function:



14.94.3.2 `reply_and_wait()` [1/2]

```

l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    long proto = 0 ) [inline]
  
```

Do an IPC reply and wait.

Parameters

in, out	<i>src_dst</i>	Input: the destination for the send operation. Output: the source of the received message.
	<i>proto</i>	Protocol to use.

Returns

the result dope of the IPC operation.

This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 878 of file [ipc_stream](#).

References [L4_IPC_SEND_TIMEOUT_0](#).

14.94.3.3 `reply_and_wait()` [2/2]

```

l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    l4_timeout_t timeout,
    long proto = 0 ) [inline]
  
```

Do an IPC reply and wait.

Parameters

<code>in, out</code>	<code>src_dst</code>	Input: the destination for the send operation. Output: the source of the received message.
	<code>timeout</code>	Timeout used for IPC.
	<code>proto</code>	Protocol to use.

Returns

the result of the IPC operation.

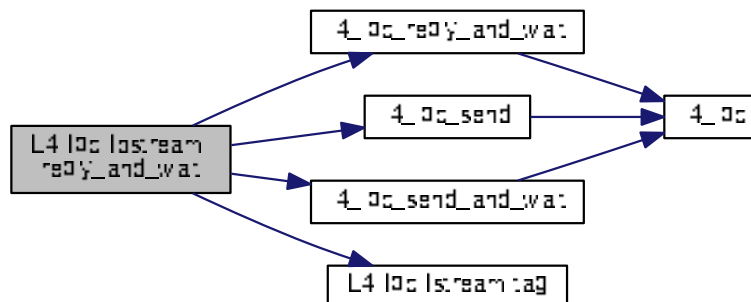
This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 979 of file [ipc_stream](#).

References [L4_INVALID_CAP](#), [l4_ipc_reply_and_wait\(\)](#), [l4_ipc_send\(\)](#), [l4_ipc_send_and_wait\(\)](#), [L4_SYSF_REPLY](#), and [L4::lpc::Iostream::tag\(\)](#).

Here is the call graph for this function:



14.94.3.4 reset()

```
void L4::lpc::Iostream::reset ( ) [inline]
```

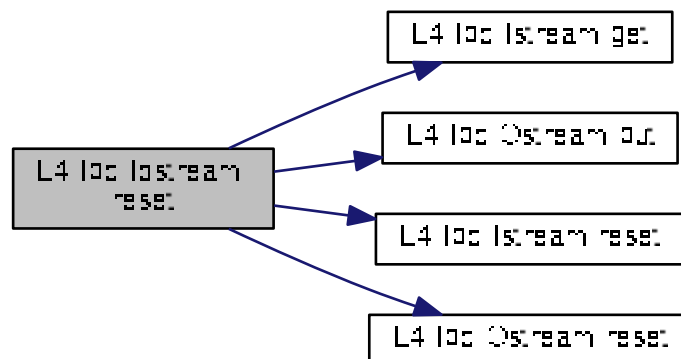
Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 819 of file [ipc_stream](#).

References [L4::lpc::Iostream::get\(\)](#), [L4::lpc::Ostream::put\(\)](#), [L4::lpc::Iostream::reset\(\)](#), and [L4::lpc::Ostream::reset\(\)](#).

Here is the call graph for this function:



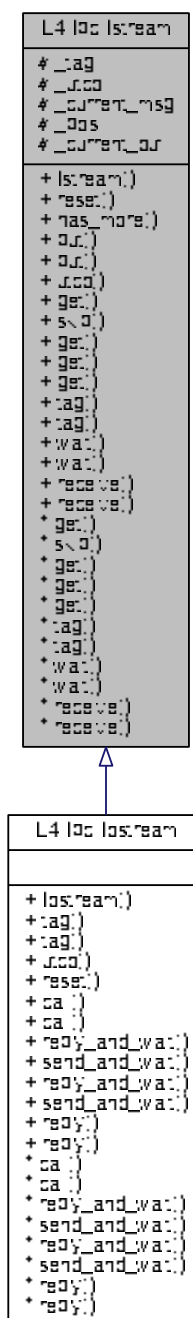
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

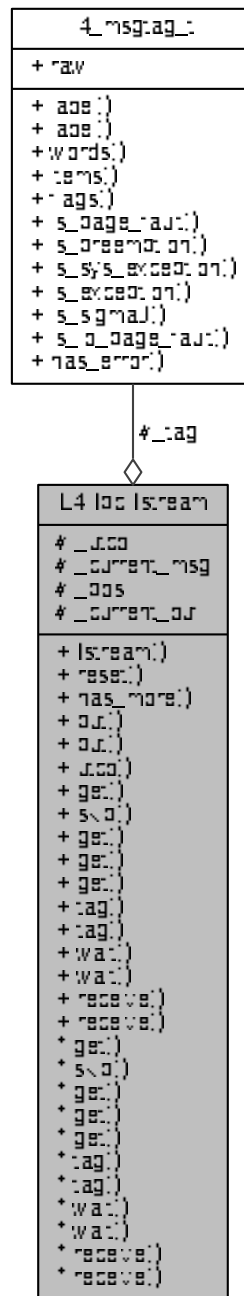
14.95 L4::ipc::Istream Class Reference

Input stream for IPC unmarshalling.

Inheritance diagram for L4::lpc::Istream:



Collaboration diagram for L4::lpc::Istream:



Public Member Functions

- `Istream (l4_utcb_t *utcb)`
Create an input stream for the given message buffer.
- `void reset ()`
Reset the stream to empty, and ready for `receive()/wait()`.

- `template<typename T >`
`bool has_more (unsigned long count=1)`
Check whether a value of type T can be obtained from the stream.
- `l4_utcb_t * utcb () const`
Return utcb pointer.

Get/Put Functions.

- `template<typename T >`
`unsigned long get (T *buf, unsigned long elems)`
Copy out an array of type T with size elements.
- `template<typename T >`
`void skip (unsigned long elems)`
Skip size elements of type T in the stream.
- `template<typename T >`
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`
Read one size elements of type T from the stream and return a pointer.
- `template<typename T >`
`bool get (T &v)`
Extract a single element of type T from the stream.
- `bool get (lpc::Varg *va)`
- `l4_msgtag_t tag () const`
Get the message tag of a received IPC.
- `l4_msgtag_t & tag ()`
Get the message tag of a received IPC.

IPC operations.

- `l4_msgtag_t wait (l4_umword_t *src)`
Wait for an incoming message from any sender.
- `l4_msgtag_t wait (l4_umword_t *src, l4_timeout_t timeout)`
Wait for an incoming message from any sender.
- `l4_msgtag_t receive (l4_cap_idx_t src)`
Wait for a message from the specified sender.
- `l4_msgtag_t receive (l4_cap_idx_t src, l4_timeout_t timeout)`

14.95.1 Detailed Description

Input stream for IPC unmarshalling.

[lpc::Istream](#) is part of the dynamic IPC marshalling infrastructure, as well as [lpc::Ostream](#) and [lpc::Iostream](#).

[lpc::Istream](#) is an input stream supporting extraction of values from an IPC message buffer. A received IPC message can be unmarshalled using the usual extraction operator (>>).

There exist some special wrapper classes to extract arrays (see `lpc_buf_cp_in` and `lpc_buf_in`) and indirect strings (see `Msg_in_buffer` and `Msg_io_buffer`).

Definition at line 347 of file [ipc_stream](#).

14.95.2 Constructor & Destructor Documentation

14.95.2.1 Istream()

```
L4::Ipc::Istream::Istream (
    l4_utcb_t * utcb ) [inline]
```

Create an input stream for the given message buffer.

The given message buffer is used for IPC operations [wait\(\)](#)/[receive\(\)](#) and received data can be extracted using the `>>` operator afterwards. In the case of indirect message parts a buffer of type `Msg_in_buffer` must be inserted into the stream before the IPC operation and contains received data afterwards.

Parameters

<i>utcb</i>	The message buffer to receive IPC messages.
-------------	---

Definition at line [361](#) of file [ipc_stream](#).

14.95.3 Member Function Documentation

14.95.3.1 get() [1/3]

```
template<typename T >
unsigned long L4::Ipc::Istream::get (
    T * buf,
    unsigned long elems ) [inline]
```

Copy out an array of type `T` with `size` elements.

Parameters

<i>buf</i>	Pointer to a buffer for size elements of type <code>T</code> .
<i>elems</i>	Number of elements of type <code>T</code> to copy out.

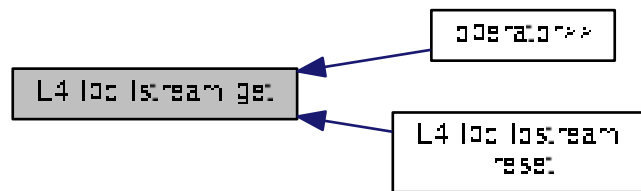
See [Istream::operator>>\(\)](#)

Definition at line [405](#) of file [ipc_stream](#).

References [L4_UNLIKELY](#).

Referenced by [operator>>\(\)](#), and [L4::Ipc::Istream::reset\(\)](#).

Here is the caller graph for this function:



14.95.3.2 `get()` [2/3]

```
template<typename T >
unsigned long L4::lpc::Istream::get (
    Msg_ptr< T > const & buf,
    unsigned long elems = 1 ) [inline]
```

Read one size elements of type T from the stream and return a pointer.

Parameters

<i>buf</i>	A Msg_ptr that is actually set to point to the element in the stream.
<i>elems</i>	Number of elements to extract (default is 1).

In contrast to a normal `get`, this version does actually not copy the data but returns a pointer to the data.

See [Istream::operator>>\(\)](#)

Definition at line 447 of file [ipc_stream](#).

References [L4_UNLIKELY](#).

14.95.3.3 `get()` [3/3]

```
template<typename T >
bool L4::lpc::Istream::get (
    T & v ) [inline]
```

Extract a single element of type T from the stream.

Parameters

out	v	The element.
-----	---	--------------

See [lstream::operator>>\(\)](#)

Definition at line 469 of file [ipc_stream](#).

References [L4_UNLIKELY](#), and [L4::lpc::msg_ptr\(\)](#).

Here is the call graph for this function:



14.95.3.4 receive()

```
l4_msgtag_t L4::Ipc::Istream::receive (
    l4_cap_idx_t src ) [inline]
```

Wait for a message from the specified sender.

Parameters

src	The sender id to receive from.
-----	--------------------------------

Returns

The IPC result code ([l4_msgtag_t](#)).

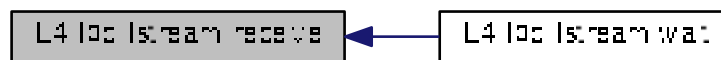
This is commonly known as 'closed wait'.

Definition at line 577 of file [ipc_stream](#).

References [L4_IPC_NEVER](#).

Referenced by [wait\(\)](#).

Here is the caller graph for this function:



14.95.3.5 reset()

```
void L4::Ipc::Istream::reset ( ) [inline]
```

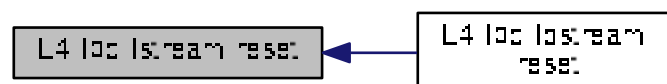
Reset the stream to empty, and ready for [receive\(\)](#)/[wait\(\)](#).

The stream is reset to the same state as on its creation.

Definition at line [371](#) of file [ipc_stream](#).

Referenced by [L4::Ipc::Iostream::reset\(\)](#).

Here is the caller graph for this function:



14.95.3.6 skip()

```
template<typename T >
void L4::Ipc::Istream::skip (
    unsigned long elems ) [inline]
```

Skip size elements of type T in the stream.

Parameters

<i>elems</i>	Number of elements to skip.
--------------	-----------------------------

Definition at line 424 of file [ipc_stream](#).

References [L4_UNLIKELY](#).

14.95.3.7 tag() [1/2]

```
l4_msgtag_t L4::Ipc::Istream::tag ( ) const [inline]
```

Get the message tag of a received IPC.

Returns

The [L4](#) message tag for the received IPC.

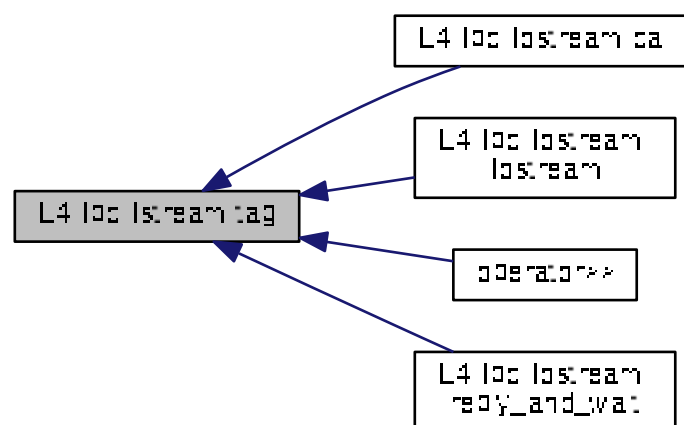
This is in particular useful for handling page faults or exceptions.

See [Istream::operator>>\(\)](#)

Definition at line 510 of file [ipc_stream](#).

Referenced by [L4::Ipc::Istream::call\(\)](#), [L4::Ipc::Istream::Istream\(\)](#), [operator>>\(\)](#), and [L4::Ipc::Istream::reply_and_wait\(\)](#).

Here is the caller graph for this function:



14.95.3.8 tag() [2/2]

```
l4_msgtag_t& L4::Ipc::Istream::tag ( ) [inline]
```

Get the message tag of a received IPC.

Returns

A reference to the [L4](#) message tag for the received IPC.

This is in particular useful for handling page faults or exceptions.

See [Istream::operator>>\(\)](#)

Definition at line [522](#) of file [ipc_stream](#).

14.95.3.9 wait() [1/2]

```
l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_t * src ) [inline]
```

Wait for an incoming message from any sender.

Parameters

out	src	Contains the sender after a successful IPC operation.
-----	-----	---

Returns

Syscall return tag.

This wait is actually known as 'open wait'.

Definition at line [553](#) of file [ipc_stream](#).

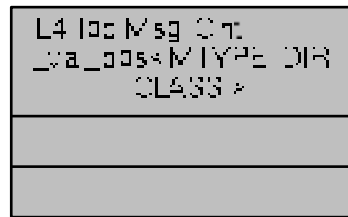
References [L4_IPC_NEVER](#).

14.95.3.10 wait() [2/2]

```
l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_t * src,
    l4_timeout_t timeout ) [inline]
```

Wait for an incoming message from any sender.

Collaboration diagram for L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS >:



14.96.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS >
```

Defines client-side handling of 'MTYPE' as RPC argument.

Template Parameters

<i>MTYPE</i>	Elem<T>::arg_type (where T is the type used in the RPC definition)
<i>DIR</i>	Dir_in (client -> server), or Dir_out (server -> client)
<i>CLASS</i>	Cls_data , Cls_item , or Cls_buffer

Definition at line 221 of file [ipc_basics](#).

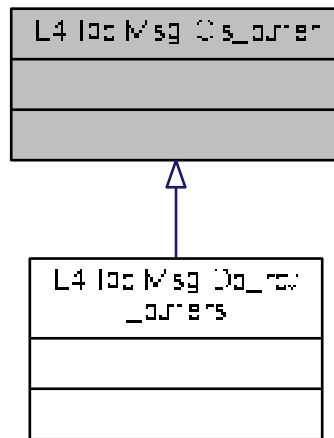
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

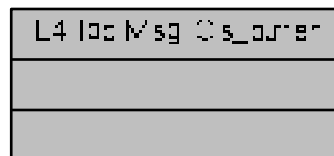
14.97 L4::lpc::Msg::Cls_buffer Struct Reference

Marker type for receive buffer values.

Inheritance diagram for L4::ipc::Msg::Cls_buffer:



Collaboration diagram for L4::ipc::Msg::Cls_buffer:



14.97.1 Detailed Description

Marker type for receive buffer values.

Definition at line 165 of file [ipc_basics](#).

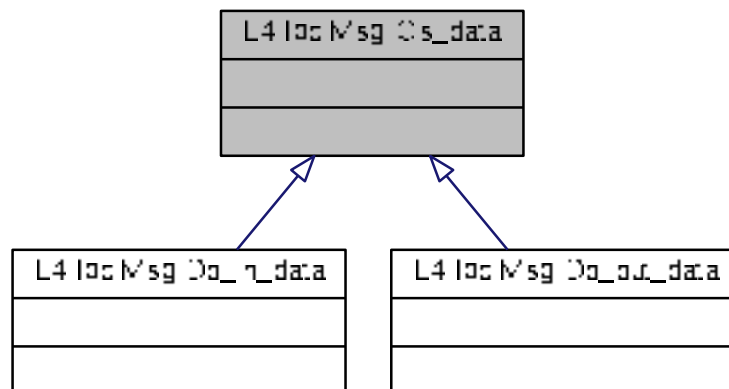
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

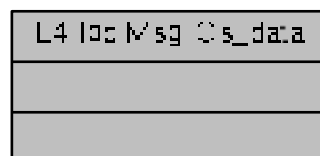
14.98 L4::ipc::Msg::Cls_data Struct Reference

Marker type for data values.

Inheritance diagram for L4::ipc::Msg::Cls_data:



Collaboration diagram for L4::ipc::Msg::Cls_data:



14.98.1 Detailed Description

Marker type for data values.

Definition at line 161 of file [ipc_basics](#).

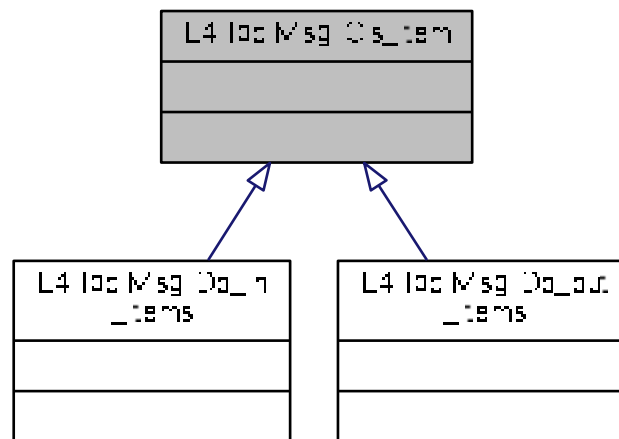
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

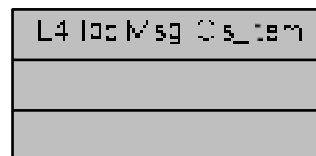
14.99 L4::ipc::Msg::Cls_item Struct Reference

Marker type for item values.

Inheritance diagram for L4::ipc::Msg::Cls_item:



Collaboration diagram for L4::ipc::Msg::Cls_item:



14.99.1 Detailed Description

Marker type for item values.

Definition at line 163 of file [ipc_basics](#).

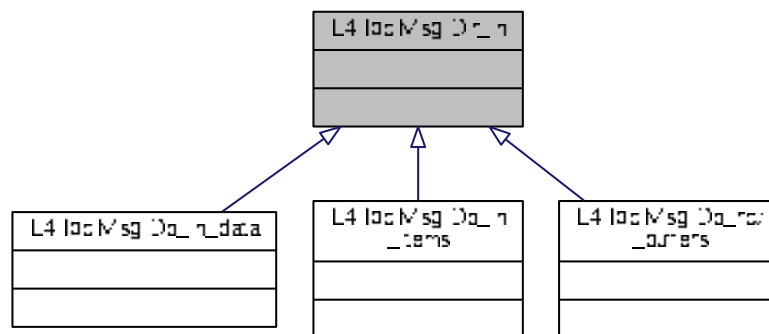
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

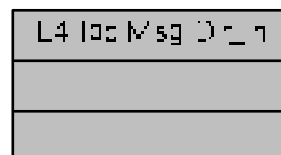
14.100 L4::ipc::Msg::Dir_in Struct Reference

Marker type for input values.

Inheritance diagram for L4::ipc::Msg::Dir_in:



Collaboration diagram for L4::ipc::Msg::Dir_in:



14.100.1 Detailed Description

Marker type for input values.

Definition at line 156 of file [ipc_basics](#).

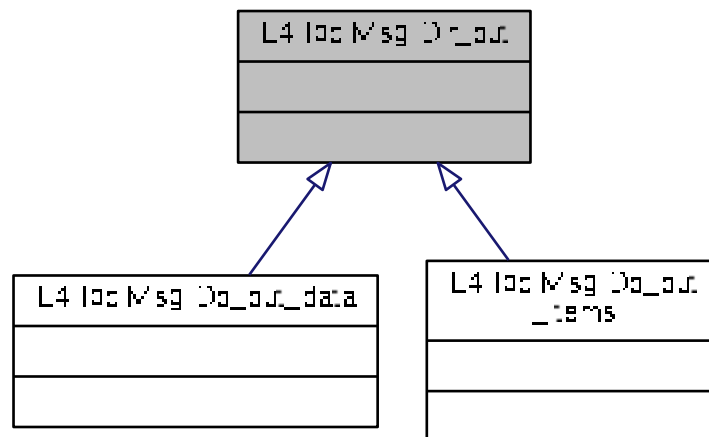
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

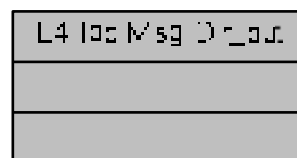
14.101 L4::ipc::Msg::Dir_out Struct Reference

Marker type for output values.

Inheritance diagram for L4::ipc::Msg::Dir_out:



Collaboration diagram for L4::ipc::Msg::Dir_out:



14.101.1 Detailed Description

Marker type for output values.

Definition at line 158 of file [ipc_basics](#).

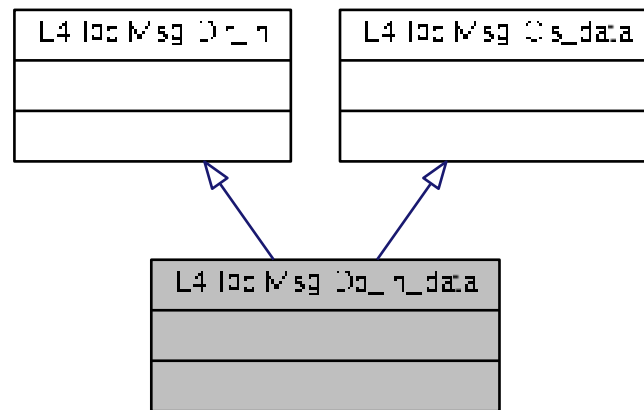
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

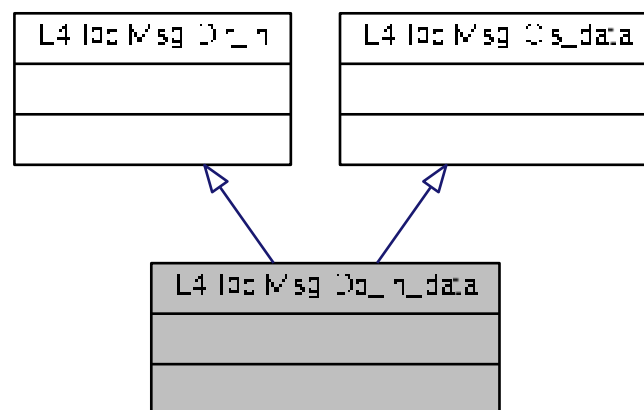
14.102 L4::lpc::Msg::Do_in_data Struct Reference

Marker for Input data.

Inheritance diagram for L4::lpc::Msg::Do_in_data:



Collaboration diagram for L4::lpc::Msg::Do_in_data:



14.102.1 Detailed Description

Marker for Input data.

Definition at line 169 of file [ipc_basics](#).

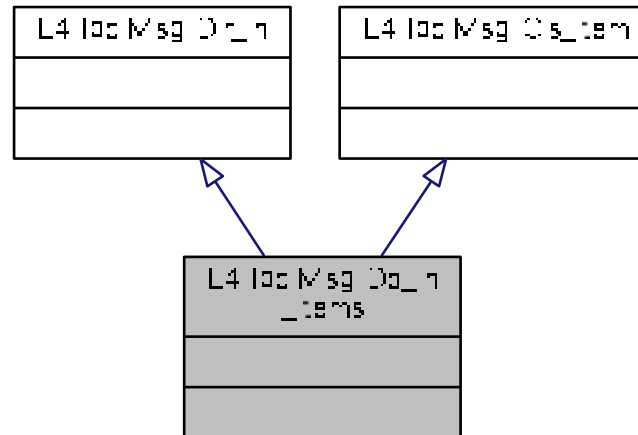
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

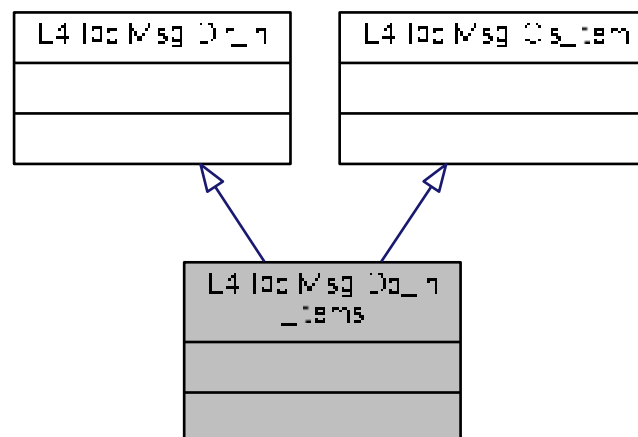
14.103 L4::lpc::Msg::Do_in_items Struct Reference

Marker for Input items.

Inheritance diagram for L4::lpc::Msg::Do_in_items:



Collaboration diagram for L4::lpc::Msg::Do_in_items:



14.103.1 Detailed Description

Marker for Input items.

Definition at line 173 of file [ipc_basics](#).

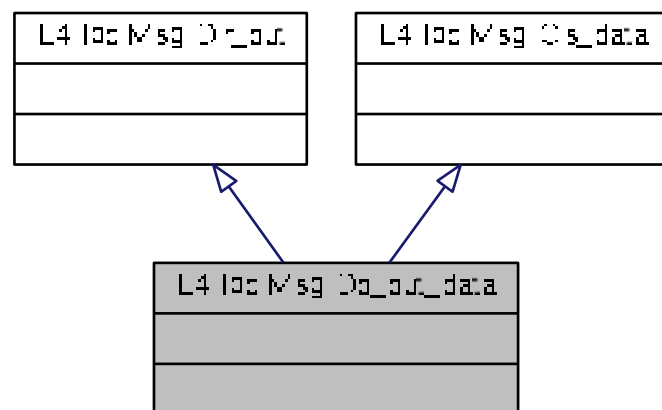
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

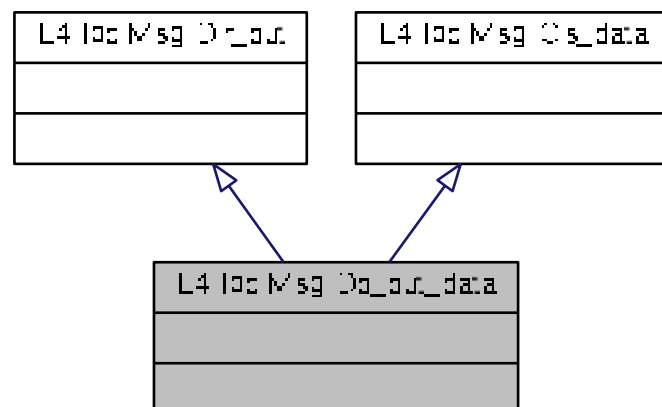
14.104 L4::lpc::Msg::Do_out_data Struct Reference

Marker for Output data.

Inheritance diagram for L4::lpc::Msg::Do_out_data:



Collaboration diagram for L4::lpc::Msg::Do_out_data:



14.104.1 Detailed Description

Marker for Output data.

Definition at line 171 of file [ipc_basics](#).

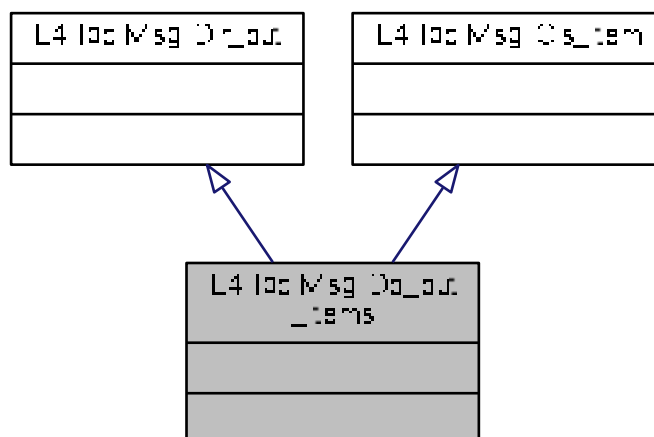
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

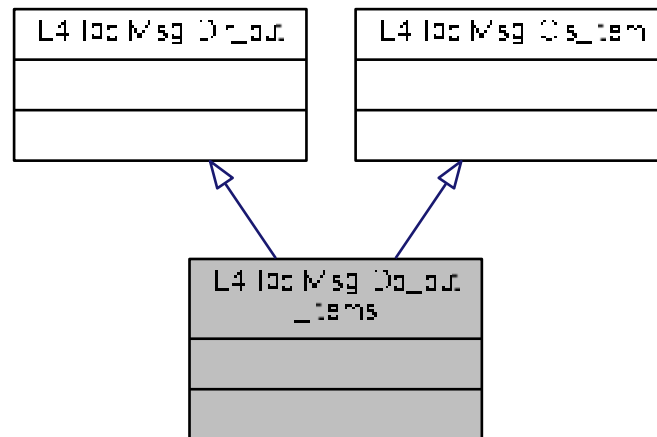
14.105 L4::lpc::Msg::Do_out_items Struct Reference

Marker for Output items.

Inheritance diagram for L4::lpc::Msg::Do_out_items:



Collaboration diagram for L4::lpc::Msg::Do_out_items:



14.105.1 Detailed Description

Marker for Output items.

Definition at line 175 of file [ipc_basics](#).

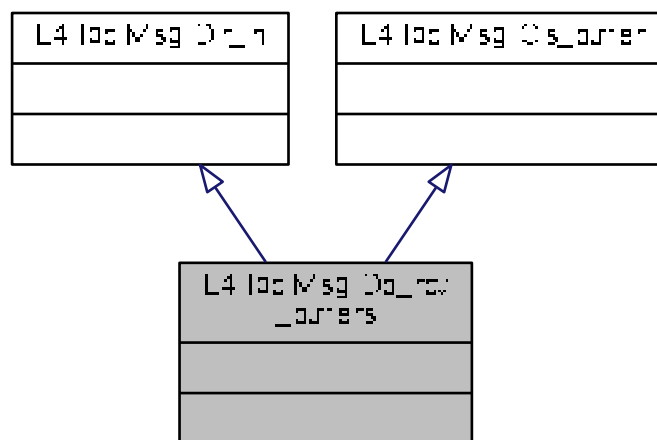
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

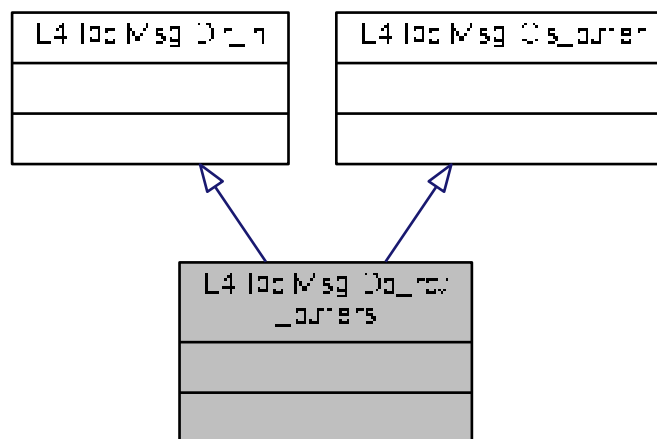
14.106 L4::lpc::Msg::Do_rcv_buffers Struct Reference

Marker for receive buffers.

Inheritance diagram for L4::ipc::Msg::Do_rcv_buffers:



Collaboration diagram for L4::ipc::Msg::Do_rcv_buffers:



14.106.1 Detailed Description

Marker for receive buffers.

Definition at line 177 of file [ipc_basics](#).

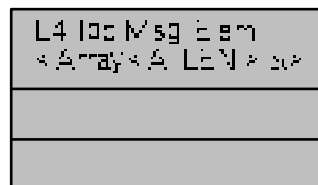
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

14.107 L4::lpc::Msg::Elem< Array< A, LEN > &> Struct Template Reference

[Array](#) as output argument.

Collaboration diagram for L4::lpc::Msg::Elem< Array< A, LEN > &>:



Public Types

- typedef [Array](#)< A, LEN > & [arg_type](#)
Array<> & at the interface.
- typedef [Array_ref](#)< A, LEN > [svr_type](#)
Array_ref<> as server storage type.
- typedef [svr_type](#) & [svr_arg_type](#)
Array_ref<> & at the server side.

14.107.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > &>
```

[Array](#) as output argument.

Definition at line 167 of file [ipc_array](#).

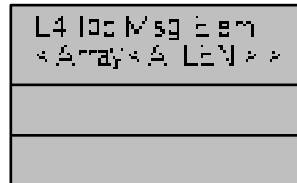
The documentation for this struct was generated from the following file:

- I4/sys/cxx/ipc_array

14.108 L4::ipc::Msg::Elem< Array< A, LEN > > Struct Template Reference

[Array](#) as input arguments.

Collaboration diagram for L4::ipc::Msg::Elem< Array< A, LEN > >:



Public Types

- typedef [Array](#)< A, LEN > [arg_type](#)
Array<> as argument at the interface.
- typedef [Array_ref](#)< A, LEN > [svr_type](#)
Array_ref<> at the server side.

14.108.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::ipc::Msg::Elem< Array< A, LEN > >
```

[Array](#) as input arguments.

Definition at line [155](#) of file [ipc_array](#).

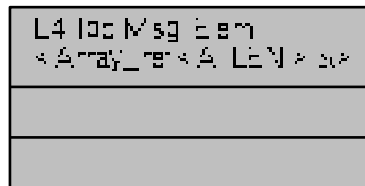
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_array`

14.109 L4::lpc::Msg::Elem< Array_ref< A, LEN > &> Struct Template Reference

[Array_ref](#) as output argument.

Collaboration diagram for L4::lpc::Msg::Elem< Array_ref< A, LEN > &>:



Public Types

- typedef [Array_ref](#)< A, LEN > & [arg_type](#)
Array_ref<> at the interface.
- typedef [Array_ref](#)< typename L4::Types::Remove_const< A >::type, LEN > [svr_type](#)
Array_ref<> as server storage.
- typedef [svr_type](#) & [svr_arg_type](#)
Array_ref<> & as server argument.

14.109.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array_ref< A, LEN > &>
```

[Array_ref](#) as output argument.

Definition at line 180 of file [ipc_array](#).

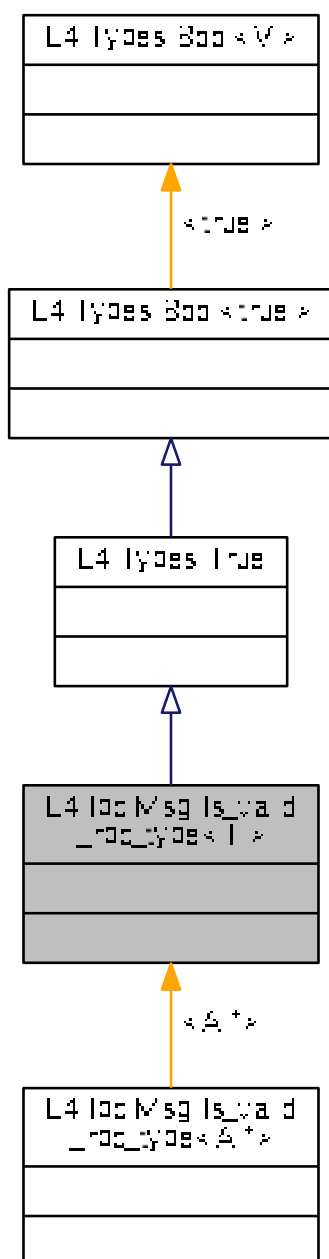
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

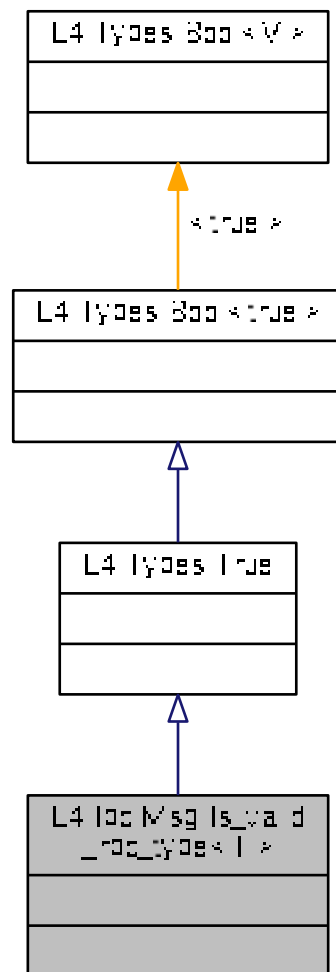
14.110 L4::lpc::Msg::ls_valid_rpc_type< T > Struct Template Reference

Type trait defining a valid RPC parameter type.

Inheritance diagram for L4::lpc::Msg::ls_valid_rpc_type< T >:



Collaboration diagram for L4::lpc::Msg::ls_valid_rpc_type< T >:



Additional Inherited Members

14.110.1 Detailed Description

```
template<typename T>
struct L4::lpc::Msg::ls_valid_rpc_type< T >
```

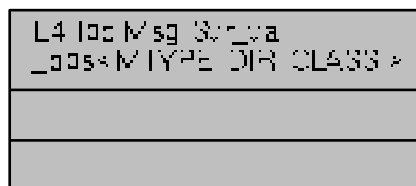
Type trait defining a valid RPC parameter type.

Definition at line 350 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

Collaboration diagram for L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >:



14.112.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >
```

Defines server-side handling for *MTYPE* server arguments.

Template Parameters

<i>MTYPE</i>	Elem<T>::svr_type (where T is the type used in the RPC definition)
<i>DIR</i>	Dir_in (client -> server), or Dir_out (server -> client)
<i>CLASS</i>	Cls_data , Cls_item , or Cls_buffer

Definition at line 275 of file [ipc_basics](#).

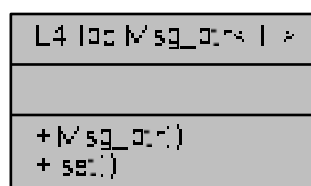
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

14.113 L4::lpc::Msg_ptr< T > Class Template Reference

Pointer to an element of type T in an [lpc::lstream](#).

Collaboration diagram for L4::lpc::Msg_ptr< T >:



Public Member Functions

- [Msg_ptr](#) (T *&p)

Create a [Msg_ptr](#) object that set pointer *p* to point into the message buffer.

14.113.1 Detailed Description

```
template<typename T>
class L4::Ipc::Msg_ptr< T >
```

Pointer to an element of type *T* in an [Ipc::Istream](#).

This wrapper can be used to extract an element of type *T* from an [Ipc::Istream](#), whereas the data is not copied out, but a pointer into the message buffer itself is returned. With is mechanism it is possible to avoid an extra copy of large data structures from a received IPC message, instead the returned pointer gives direct access to the data in the message.

See [msg_ptr\(\)](#).

Definition at line 242 of file [ipc_stream](#).

14.113.2 Constructor & Destructor Documentation

14.113.2.1 Msg_ptr()

```
template<typename T>
L4::Ipc::Msg_ptr< T >::Msg_ptr (
    T *& p ) [inline], [explicit]
```

Create a [Msg_ptr](#) object that set pointer *p* to point into the message buffer.

Parameters

<i>p</i>	The pointer that is adjusted to point into the message buffer.
----------	--

Definition at line 253 of file [ipc_stream](#).

The documentation for this class was generated from the following file:

- [I4/cxx/ipc_stream](#)

14.114 L4::Ipc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

Collaboration diagram for L4::lpc::Opt< T >:

L4::lpc::Opt< T >
+ _value + _valid
+ Opt() + Opt() + operator=() + set_valid() + operator*() + operator*() + value() + value() + s_valid()

Public Member Functions

- [Opt \(\)](#)
Make an absent optional argument.
- [Opt \(T value\)](#)
Make a present optional argument with the given value.
- [Opt & operator= \(T value\)](#)
Assign a value to the optional argument (makes the argument present)
- void [set_valid](#) (bool valid=true)
Set the argument to present or absent.
- T * [operator-> \(\)](#)
Get the pointer to the value.
- T const * [operator-> \(\) const](#)
Get the const pointer to the value.
- T [value \(\) const](#)
Get the value.
- T & [value \(\)](#)
Get the value.
- bool [is_valid \(\) const](#)
Get true if present, false if not.

Data Fields

- T [_value](#)
The value.
- bool [_valid](#)
True if the optional argument is present, false else.

14.114.1 Detailed Description

```
template<typename T>  
struct L4::ipc::Opt< T >
```

Attribute for defining an optional RPC argument.

Definition at line [147](#) of file [ipc_types](#).

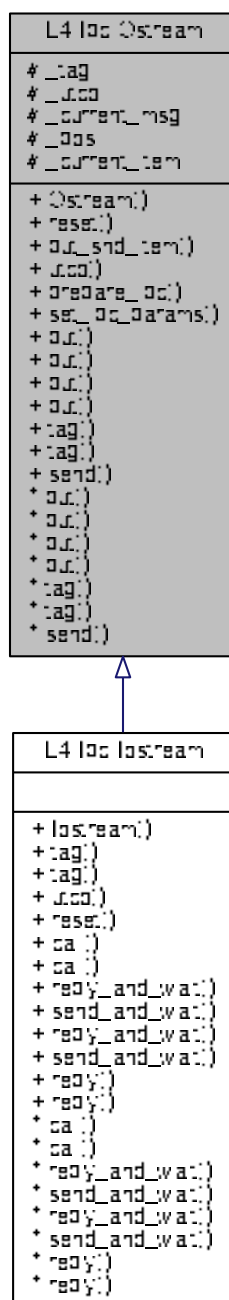
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

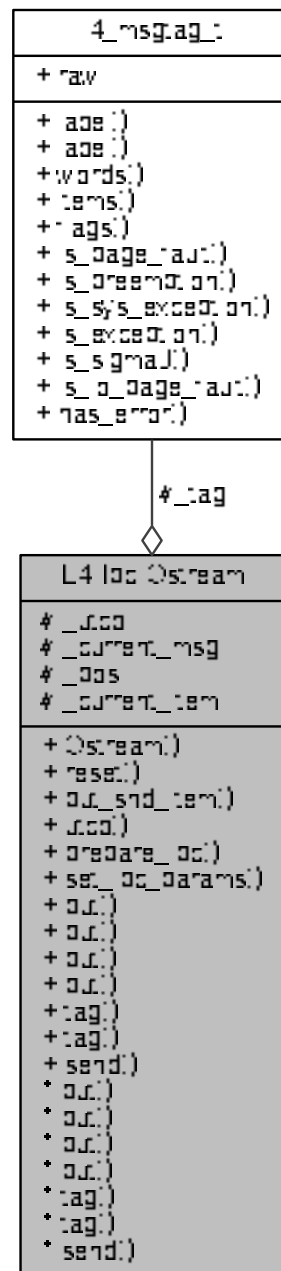
14.115 L4::ipc::Ostream Class Reference

Output stream for IPC marshalling.

Inheritance diagram for L4::lpc::Ostream:



Collaboration diagram for L4::ipc::Ostream:



Public Member Functions

- [Ostream](#) ([l4_utcb_t](#) *utcb)
Create an IPC output stream using the given message buffer *utcb*.
- void [reset](#) ()
Reset the stream to empty, same state as a newly created stream.
- [l4_utcb_t](#) * [utcb](#) () const

Return utcb pointer.

Get/Put functions.

These functions are basically used to implement the insertion operators (<<) and should not be called directly.

- `template<typename T >`
`bool put (T *buf, unsigned long size)`
Put an array with `size` elements of type `T` into the stream.
- `template<typename T >`
`bool put (T const &v)`
Insert an element of type `T` into the stream.
- `int put (Varg const &va)`
- `template<typename T >`
`int put (Varg_t< T > const &va)`
- `l4_msgtag_t tag () const`
Extract the `L4` message tag from the stream.
- `l4_msgtag_t & tag ()`
Extract a reference to the `L4` message tag from the stream.

IPC operations.

- `l4_msgtag_t send (l4_cap_idx_t dst, long proto=0, unsigned flags=0)`
Send the message via IPC to the given receiver.

14.115.1 Detailed Description

Output stream for IPC marshalling.

`lpc::Ostream` is part of the dynamic IPC marshalling infrastructure, as well as `lpc::Istream` and `lpc::Iostream`.

`lpc::Ostream` is an output stream supporting insertion of values into an IPC message buffer. A IPC message can be marshalled using the usual insertion operator <<, see [IPC stream operators](#) .

There exist some special wrapper classes to insert arrays (see `lpc::Buf_cp_out`) and indirect strings (see `Msg_↔_out_buffer` and `Msg_io_buffer`).

Definition at line 625 of file `ipc_stream`.

14.115.2 Member Function Documentation

14.115.2.1 put() [1/2]

```
template<typename T >
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size ) [inline]
```

Put an array with `size` elements of type `T` into the stream.

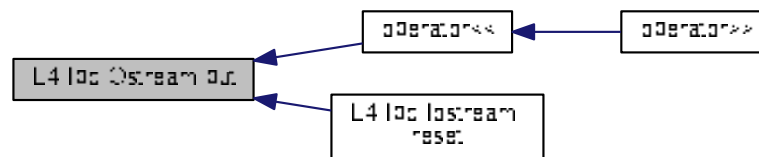
Parameters

<i>buf</i>	A pointer to the array to insert into the buffer.
<i>size</i>	The number of elements in the array.

Definition at line 662 of file [ipc_stream](#).

Referenced by [operator<<\(\)](#), and [L4::ipc::loststream::reset\(\)](#).

Here is the caller graph for this function:

14.115.2.2 `put()` [2/2]

```
template<typename T >
bool L4::IpC::Ostream::put (
    T const & v ) [inline]
```

Insert an element of type `T` into the stream.

Parameters

<i>v</i>	The element to insert.
----------	------------------------

Definition at line 680 of file [ipc_stream](#).

14.115.2.3 `send()`

```
l4_msgtag_t L4::IpC::Ostream::send (
    l4_cap_idx_t dst,
    long proto = 0,
    unsigned flags = 0 ) [inline]
```

Send the message via IPC to the given receiver.

Parameters

<i>dst</i>	The destination for the message.
<i>proto</i>	Protocol to use.
<i>flags</i>	Flags to use.

Returns

Syscall return tag.

Definition at line 957 of file [ipc_stream](#).

References [L4_IPC_NEVER](#), [l4_ipc_send\(\)](#), and [L4_MSGTAG_FLAGS](#).

Here is the call graph for this function:

14.115.2.4 `tag()` [1/2]

```
l4_msgtag_t L4::Ipc::Ostream::tag ( ) const [inline]
```

Extract the [L4](#) message tag from the stream.

Returns

the extracted [L4](#) message tag.

Definition at line 708 of file [ipc_stream](#).

Referenced by [operator<<\(\)](#).

Here is the caller graph for this function:



14.115.2.5 `tag()` [2/2]

```
l4_msgtag_t& L4::Ipc::Ostream::tag ( ) [inline]
```

Extract a reference to the [L4](#) message tag from the stream.

Returns

A reference to the [L4](#) message tag.

Definition at line [715](#) of file [ipc_stream](#).

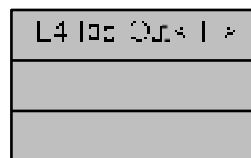
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

14.116 L4::lpc::Out< T > Struct Template Reference

Mark an argument as a output value in an RPC signature.

Collaboration diagram for L4::lpc::Out< T >:

**14.116.1 Detailed Description**

```
template<typename T>
struct L4::lpc::Out< T >
```

Mark an argument as a output value in an RPC signature.

Template Parameters

<i>T</i>	The original type of the argument.
----------	------------------------------------

Note

The use of Out<> is usually not needed, because typical out-put data types in C++ (pointers to non-const objects or non-const references are interpreted as output values anyway. However, there are some data types, such as returned capabilities that can be marked as such by using Out<>.

Definition at line 42 of file [ipc_types](#).

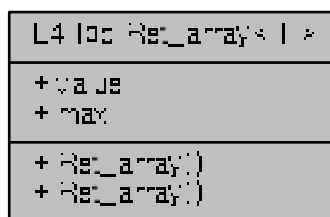
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.117 L4::lpc::Ret_array< T > Struct Template Reference

Dynamically sized output array of type T.

Collaboration diagram for L4::lpc::Ret_array< T >:



14.117.1 Detailed Description

```
template<typename T>
struct L4::lpc::Ret_array< T >
```

Dynamically sized output array of type T.

Template Parameters

<code>T</code>	The data-type of each array element.
----------------	--------------------------------------

`Ret_array<>` is a special dynamically sized output array where the number of transmitted elements is passed in the return value of the call (if positive)

Definition at line 34 of file [ipc_ret_array](#).

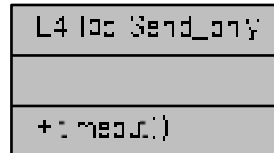
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_ret_array](#)

14.118 L4::ipc::Send_only Struct Reference

RPC attribute for a send-only RPC.

Collaboration diagram for L4::ipc::Send_only:



14.118.1 Detailed Description

RPC attribute for a send-only RPC.

This class can be used as FLAGS parameter to L4::ipc::Msg::Rpc_call and L4::ipc::Msg::Rpc_inline_call templates and declares the RPC to use send-only semantics and timeouts.

Examples:

```
L4_RPC(long, send, (unsigned value), L4::Ipс::Send_only);
```

Definition at line 263 of file [ipc_iface](#).

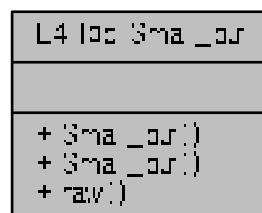
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

14.119 L4::ipc::Small_buf Class Reference

A receive item for receiving a single capability.

Collaboration diagram for L4::ipc::Small_buf:



Public Member Functions

- [Small_buf](#) ([L4::Cap](#)< void > cap, unsigned long flags=0)
Create a receive item from a C++ cap.
- [Small_buf](#) ([l4_cap_idx_t](#) cap, unsigned long flags=0)
Create a receive item from a C cap.

14.119.1 Detailed Description

A receive item for receiving a single capability.

This class is the main abstraction for receiving capabilities via [lpc::lstream](#). To receive a capability an instance of [Small_buf](#) that refers to an empty capability slot must be inserted into the [lpc::lstream](#) before the receive operation.

Definition at line 268 of file [ipc_types](#).

14.119.2 Constructor & Destructor Documentation

14.119.2.1 Small_buf() [1/2]

```
L4::Ipc::Small_buf::Small_buf (
    L4::Cap< void > cap,
    unsigned long flags = 0 ) [inline], [explicit]
```

Create a receive item from a C++ cap.

Parameters

<i>cap</i>	Capability slot where to save the capability.
<i>flags</i>	Receive buffer flags, see l4_msg_item_consts_t . L4_RCV_ITEM_SINGLE_CAP will always be set.

Definition at line 278 of file [ipc_types](#).

14.119.2.2 Small_buf() [2/2]

```
L4::Ipc::Small_buf::Small_buf (
    l4\_cap\_idx\_t cap,
    unsigned long flags = 0 ) [inline], [explicit]
```

Create a receive item from a C cap.

Parameters

<i>cap</i>	Capability slot where to save the capability.
<i>flags</i>	Receive buffer flags, see l4_msg_item_consts_t . L4_RCV_ITEM_SINGLE_CAP will always be set.

Definition at line 285 of file [ipc_types](#).

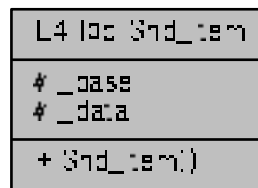
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.120 L4::lpc::Snd_item Class Reference

RPC wrapper for a send item.

Collaboration diagram for L4::lpc::Snd_item:



14.120.1 Detailed Description

RPC wrapper for a send item.

Definition at line 294 of file [ipc_types](#).

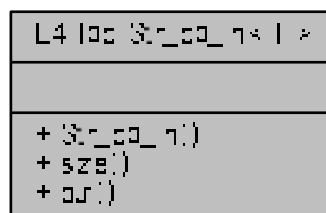
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.121 L4::lpc::Str_cp_in< T > Class Template Reference

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

Collaboration diagram for L4::lpc::Str_cp_in< T >:



Public Member Functions

- [Str_cp_in](#) (T *v, unsigned long &size)
Create a buffer for extracting an array from an [lpc::lstream](#).

14.121.1 Detailed Description

```
template<typename T>
class L4::lpc::Str_cp_in< T >
```

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

An instance of [Str_cp_in](#) can be used to extract a zero-terminated string an [lpc::lstream](#). The data from the received message is thereby copied to the given buffer and size is set to the number of characters found in the stream. The string is zero terminated in any circumstances. When the given buffer is smaller than the received string the last byte in the buffer will be the zero terminator. In the case the received string is shorter than the given buffer the zero termination will be placed behind the received data. This provides a zero-terminated result even in cases where the sender did not provide proper termination or in cases of too small receiver buffers.

See also

[str_cp_in\(\)](#).

Definition at line 191 of file [ipc_stream](#).

14.121.2 Constructor & Destructor Documentation

14.121.2.1 Str_cp_in()

```
template<typename T>
L4::lpc::Str_cp_in< T >::Str_cp_in (
    T * v,
    unsigned long & size ) [inline]
```

Create a buffer for extracting an array from an [lpc::lstream](#).

Parameters

	<i>v</i>	The buffer for string.
<i>in, out</i>	<i>size</i>	Input: The number of bytes available in <i>v</i> Output: The number of bytes received (including the terminator).

Definition at line 202 of file [ipc_stream](#).

The documentation for this class was generated from the following file:

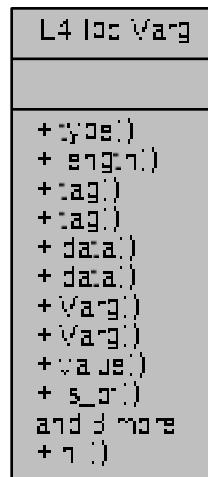
- [l4/cxx/ipc_stream](#)

14.122 L4::lpc::Varg Class Reference

Variably sized RPC argument.

Inherited by L4::lpc::Varg_t< T >.

Collaboration diagram for L4::lpc::Varg:



Public Types

- typedef [l4_umword_t](#) Tag
The data type for the tag.

Public Member Functions

- L4_varg_type [type](#) () const
- unsigned [length](#) () const
Get the size of the RPC argument.
- [Tag](#) [tag](#) () const
- void [tag](#) (Tag tag)
Set Varg tag (usually from message)
- void [data](#) (char const *d)
Set Varg to indirect data value (usually in UTCB)
- char const * [data](#) () const
- [Varg](#) ()=default
Make uninitialized Varg.
- [Varg](#) (L4_varg_type t, void const *v, int len)
Make an indirect varg.

- `template<typename V >`
`Va_type< V >::Ret_value value () const`
- `template<typename T >`
`bool is_of () const`
- `bool is_nil () const`
- `bool is_of_int () const`
- `template<typename T >`
`bool get_value (typename Va_type< T >::Value *v) const`
Get the value of the Varg as type T.
- `template<typename T >`
`void set_value (void const *d)`
Set to indirect value of type T.
- `template<typename T >`
`void set_direct_value (T val, typename L4::Types::Enable_if< sizeof(T)<=sizeof(char const *) , bool >::type=true)`
Set to directly stored value of type T.
- `template<typename T >`
`Varg (T const *data)`
Make Varg from indirect value (pointer)
- `Varg (char const *data)`
Make Varg from null-terminated string.
- `template<typename T >`
`Varg (T data, typename L4::Types::Enable_if< sizeof(T)<=sizeof(char const *) , bool >::type=true)`
Make Varg from direct value.

14.122.1 Detailed Description

Variably sized RPC argument.

Definition at line 96 of file [ipc_varg](#).

14.122.2 Member Function Documentation

14.122.2.1 data()

```
char const* L4::Ipc::Varg::data ( ) const [inline]
```

Returns

pointer to the data, also safe for direct data

Definition at line 123 of file [ipc_varg](#).

14.122.2.2 get_value()

```
template<typename T >
bool L4::Ipc::Varg::get_value (
    typename Va_type< T >::Value * v ) const [inline]
```

Get the value of the [Varg](#) as type T.

Template Parameters

<i>T</i>	The expected type of the Varg .
----------	---

Parameters

<i>v</i>	Pointer to store the value
----------	----------------------------

Returns

true when the [Varg](#) is of type *T*, false if not

Definition at line [184](#) of file [ipc_varg](#).

14.122.2.3 is_nil()

```
bool L4::Ipc::Varg::is_nil ( ) const [inline]
```

Returns

true if the [Varg](#) is of nil type.

Definition at line [171](#) of file [ipc_varg](#).

14.122.2.4 is_of()

```
template<typename T >
bool L4::Ipc::Varg::is_of ( ) const [inline]
```

Returns

true if the [Varg](#) is of type *T*

Definition at line [168](#) of file [ipc_varg](#).

14.122.2.5 is_of_int()

```
bool L4::Ipc::Varg::is_of_int ( ) const [inline]
```

Returns

true if the [Varg](#) is an integer type (signed or unsigned).

Definition at line [174](#) of file [ipc_varg](#).

14.122.2.6 length()

```
unsigned L4::Ipc::Varg::length ( ) const [inline]
```

Get the size of the RPC argument.

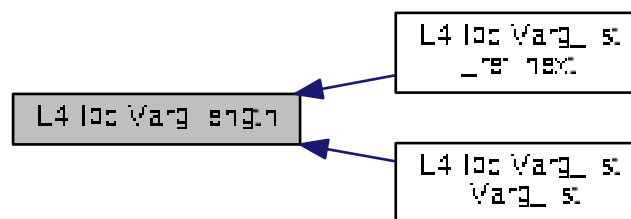
Returns

The size of the RPC argument

Definition at line 114 of file [ipc_varg](#).

Referenced by [L4::Ipc::Varg_list_ref::next\(\)](#), and [L4::Ipc::Varg_list< MAX >::Varg_list\(\)](#).

Here is the caller graph for this function:



14.122.2.7 tag()

```
Tag L4::Ipc::Varg::tag ( ) const [inline]
```

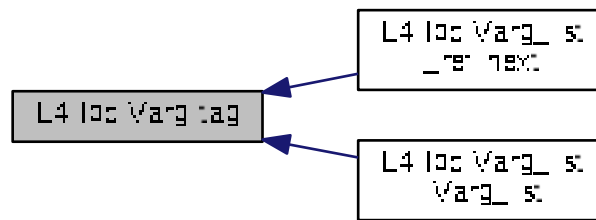
Returns

the tag value (the `Direct_data` bit masked)

Definition at line 116 of file [ipc_varg](#).

Referenced by [L4::Ipc::Varg_list_ref::next\(\)](#), and [L4::Ipc::Varg_list< MAX >::Varg_list\(\)](#).

Here is the caller graph for this function:



14.122.2.8 type()

```
L4_varg_type L4::Ipc::Varg::type ( ) const [inline]
```

Returns

the type field of the tag

Definition at line 109 of file [ipc_varg](#).

14.122.2.9 value()

```
template<typename V >
Va_type<V>::Ret_value L4::Ipc::Varg::value ( ) const [inline]
```

Template Parameters

V	The data type of the value to retrieve.
----------	---

Precondition

The [Varg](#) must be of type *V* (otherwise the result is unpredictable).

Returns

The value of the [Varg](#) as type *V*.

Definition at line 154 of file [ipc_varg](#).

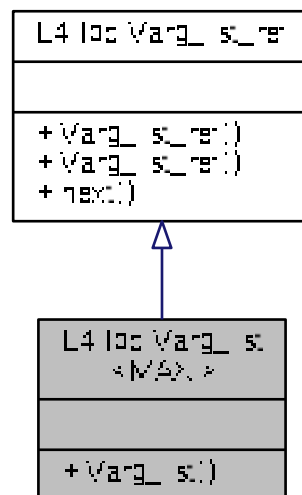
The documentation for this class was generated from the following file:

- `l4/sys/cxx/ipc_varg`

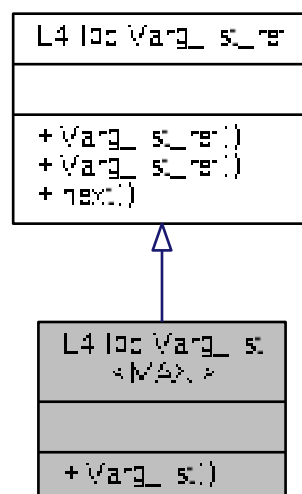
14.123 L4::lpc::Varg_list< MAX > Class Template Reference

Self-contained list of variable-sized RPC parameters.

Inheritance diagram for L4::lpc::Varg_list< MAX >:



Collaboration diagram for L4::lpc::Varg_list< MAX >:



Public Member Functions

- [Varg_list](#) ([Varg_list_ref](#) const &r)

Create a parameter list as a copy from a referencing list.

14.123.1 Detailed Description

```
template<unsigned MAX>
class L4::lpc::Varg_list< MAX >
```

Self-contained list of variable-sized RPC parameters.

Works like [Varg_list_ref](#) but contains a full copy of the data. Use this as a parameter in server functions, if the handler function needs to use the UTCB (e.g. while sending further IPC).

Definition at line 239 of file [ipc_varg](#).

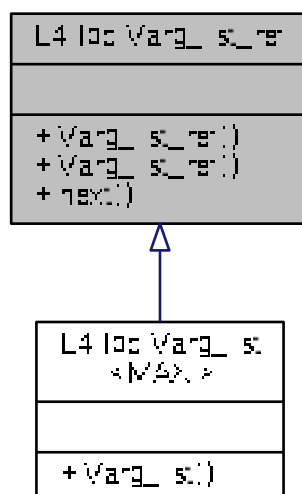
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_varg](#)

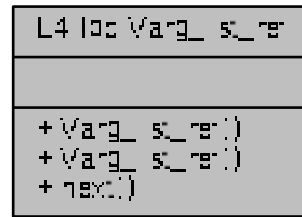
14.124 L4::lpc::Varg_list_ref Class Reference

List of variable-sized RPC parameters as received by the server.

Inheritance diagram for L4::lpc::Varg_list_ref:



Collaboration diagram for L4::Ipc::Varg_list_ref:



Public Member Functions

- [Varg_list_ref](#) ()
Create an empty parameter list.
- [Varg_list_ref](#) (void const *start, void const *end)
Create a parameter list over a given memory region.
- [Varg_next](#) ()
Get the next parameter in the list.

14.124.1 Detailed Description

List of variable-sized RPC parameters as received by the server.

The list can be traversed exactly once using [next\(\)](#).

This is a reference list, where the returned [Varg](#) point to data in the underlying storage, conventionally the UTCB. This type should only be used in server functions when the implementation can ensure that all content is read before the UTCB is reused (e.g. for IPC), otherwise use [Varg_list](#).

Definition at line 252 of file [ipc_varg](#).

14.124.2 Constructor & Destructor Documentation

14.124.2.1 Varg_list_ref()

```

L4::Ipc::Varg_list_ref::Varg_list_ref (
    void const * start,
    void const * end ) [inline]
  
```

Create a parameter list over a given memory region.

Parameters

<i>start</i>	Pointer to start of the parameter list.
<i>end</i>	Pointer to end of the list (inclusive).

Definition at line [269](#) of file [ipc_varg](#).

The documentation for this class was generated from the following file:

- `l4/sys/cxx/ipc_varg`

14.125 L4::ipc_gate Class Reference

The C++ IPC gate interface.

14.125.1 Detailed Description

The C++ IPC gate interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [L4::Factory](#) interface. [L4::ipc_gate::bind_thread\(\)](#) binds an [L4::Thread](#) as the receiver of all messages to an IPC gate.

The [bind_thread\(\)](#) call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The two least significant bits of the *label* are ORed with the L4_CAP_FPAGE_S and L4_CAP_FPAGE_W bits stored in the capability when transferred to the receiver. This means the *label* should usually have its two least significant bits set to zero. The *label* is only visible in the [L4::Task](#) which is running the thread the IPC gate was bound to and cannot be altered by the sender.

Include File

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C [IPC-Gate API](#).

Definition at line 56 of file [ipc_gate](#).

14.125.2 Member Function Documentation

14.125.2.1 get_infos()

```
l4_msgtag_t L4::Ipc_gate::get_infos (
    l4_umword_t * label )
```

Get information about the IPC-gate.

Parameters

out	<i>label</i>	The label of the IPC gate is returned here.
-----	--------------	---

Returns

System call return tag.

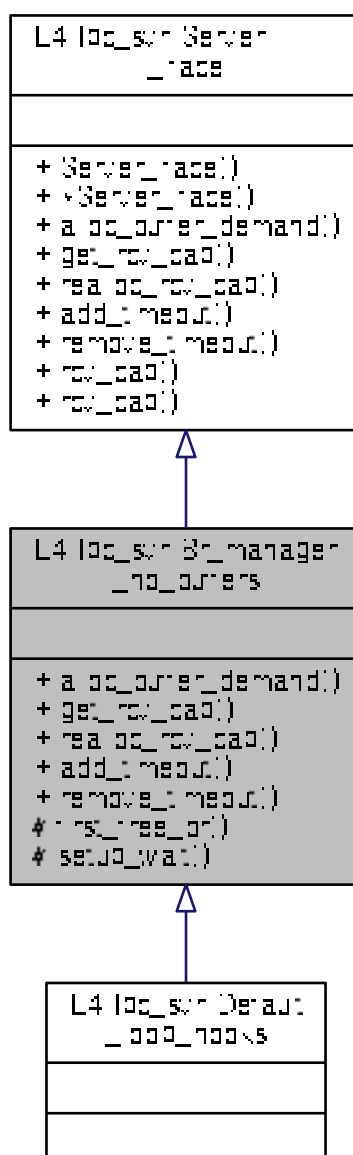
The documentation for this class was generated from the following file:

- [l4/sys/ipc_gate](#)

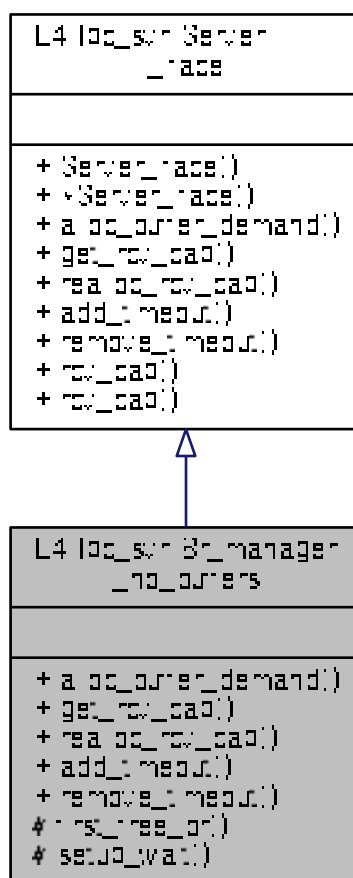
14.126 L4::ipc_svr::Br_manager_no_buffers Class Reference

Empty implementation of [Server_iface](#).

Inheritance diagram for L4::ipc_svr::Br_manager_no_buffers:



Collaboration diagram for L4::lpc_svr::Br_manager_no_buffers:



Public Member Functions

- int [alloc_buffer_demand](#) ([Demand](#) const &demand)
Tells the server to allocate buffers for the given demand.
- [L4::Cap](#)< void > [get_rcv_cap](#) (int) const
Returns [L4::Cap](#)< void> ::Invalid, we have no buffer management.
- int [realloc_rcv_cap](#) (int)
Returns -L4_ENOMEM, we have no buffer management.
- int [add_timeout](#) ([Timeout](#) *, [l4_kernel_clock_t](#))
Returns -L4_ENOSYS, we have no timeout queue.
- int [remove_timeout](#) ([Timeout](#) *)
Returns -L4_ENOSYS, we have no timeout queue.

Protected Member Functions

- unsigned [first_free_br](#) () const
Returns 1 as first free buffer.
- void [setup_wait](#) ([l4_utcb_t](#) *utcb, [L4::lpc_svr::Reply_mode](#))
Setup wait function for the server loop (Server<>).

Additional Inherited Members

14.126.1 Detailed Description

Empty implementation of [Server_iface](#).

This implementation of [Server_iface](#) provides no buffer or timeout management at all it just returns errors for all calls that express other than empty demands. However, this may be useful for very simple servers that serve simple server objects only.

Definition at line 216 of file [ipc_server_loop](#).

14.126.2 Member Function Documentation

14.126.2.1 `alloc_buffer_demand()`

```
int L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand (
    Demand const & demand ) [inline], [virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see Demand.
---------------	---

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Returns

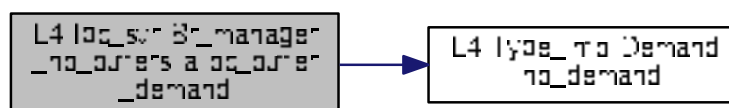
success (0) if demand is empty, -L4_ENOMEM else.

Implements [L4::Ipc_svr::Server_iface](#).

Definition at line 223 of file [ipc_server_loop](#).

References [L4_ENOMEM](#), [L4_EOK](#), and [L4::Type_info::Demand::no_demand\(\)](#).

Here is the call graph for this function:



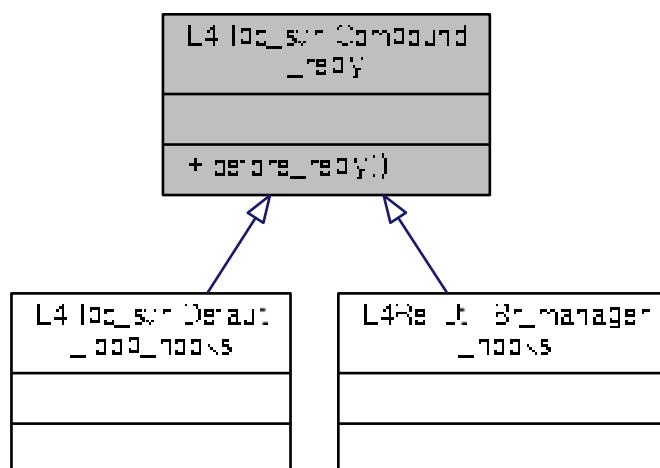
The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_server_loop

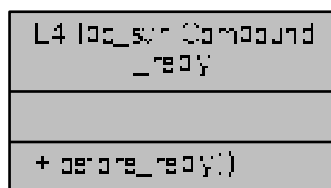
14.127 L4::lpc_svr::Compound_reply Struct Reference

Mix in for LOOP_HOOKS to always use compound reply and wait.

Inheritance diagram for L4::lpc_svr::Compound_reply:



Collaboration diagram for L4::lpc_svr::Compound_reply:



14.127.1 Detailed Description

Mix in for LOOP_HOOKS to always use compound reply and wait.

Definition at line 77 of file [ipc_server_loop](#).

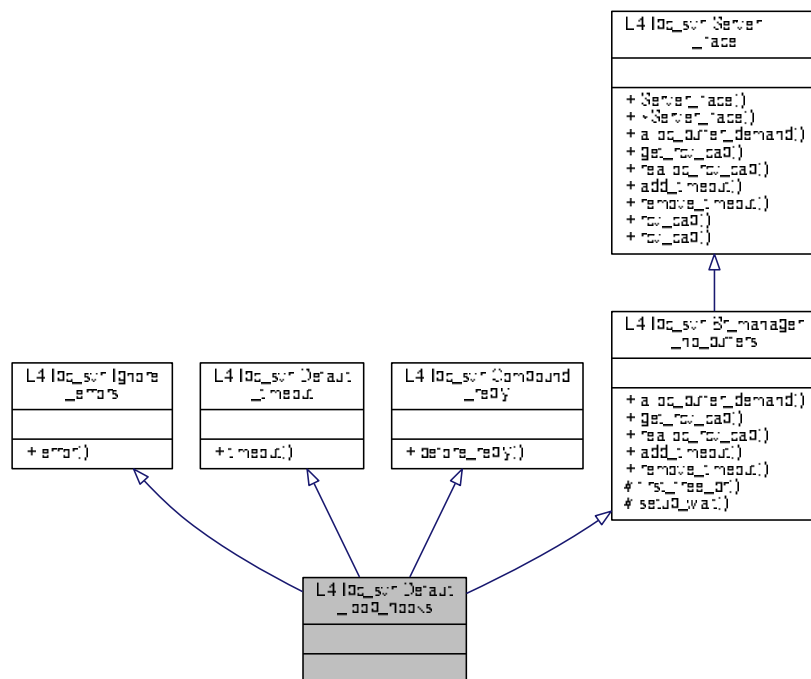
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

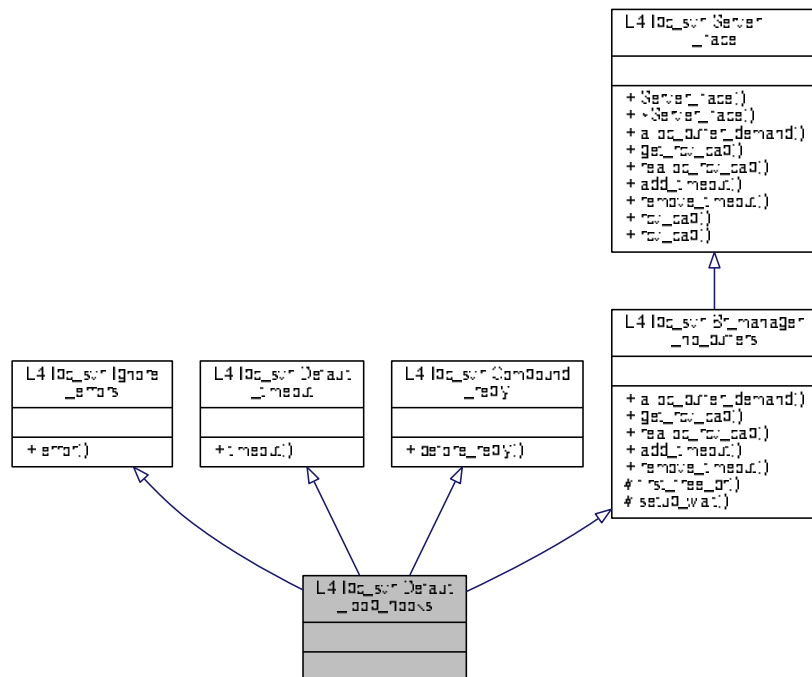
14.128 L4::lpc_svr::Default_loop_hooks Struct Reference

Default LOOP_HOOKS.

Inheritance diagram for L4::lpc_svr::Default_loop_hooks:



Collaboration diagram for L4::lpc_svr::Default_loop_hooks:



Additional Inherited Members

14.128.1 Detailed Description

Default LOOP_HOOKS.

Combination of [Ignore_errors](#), [Default_timeout](#), [Compound_reply](#), and [Br_manager_no_buffers](#).

Definition at line 268 of file [ipc_server_loop](#).

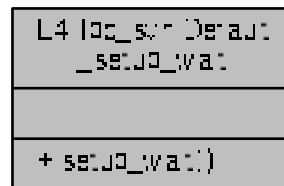
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_server_loop`

14.129 L4::lpc_svr::Default_setup_wait Struct Reference

Mix in for LOOP_HOOKS for setup_wait no op.

Collaboration diagram for L4::lpc_svr::Default_setup_wait:



14.129.1 Detailed Description

Mix in for LOOP_HOOKS for setup_wait no op.

Definition at line 88 of file [ipc_server_loop](#).

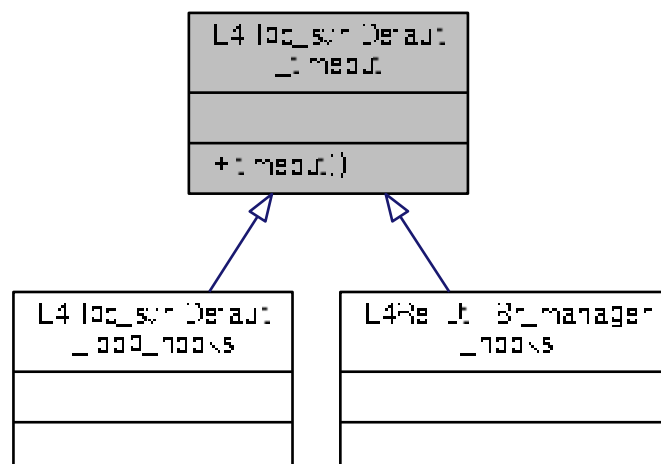
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

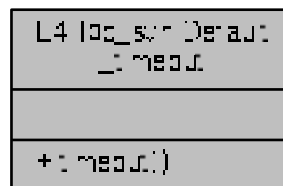
14.130 L4::lpc_svr::Default_timeout Struct Reference

Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

Inheritance diagram for L4::lpc_svr::Default_timeout:



Collaboration diagram for L4::lpc_svr::Default_timeout:



14.130.1 Detailed Description

Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

Definition at line 69 of file [ipc_server_loop](#).

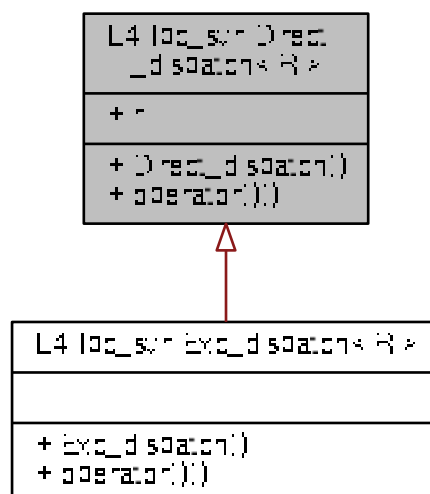
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

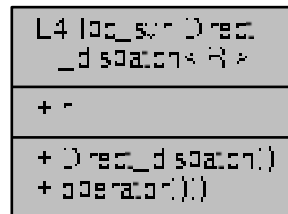
14.131 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference

Direct disptach helper, for forwarding dispatch calls a registry *R*.

Inheritance diagram for L4::lpc_svr::Direct_dispatch< R >:



Collaboration diagram for L4::lpc_svr::Direct_dispatch< R >:



Public Member Functions

- [Direct_dispatch](#) (R &r)
Make a direct dispatcher.
- [l4_msgtag_t operator\(\)](#) ([l4_msgtag_t](#) tag, [l4_umword_t](#) obj, [l4_utcb_t](#) *utcb)
call operator forwarding to r.dispatch()

Data Fields

- R & [r](#)
stores a reference to the registry object

14.131.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R >
```

Direct dispatch helper, for forwarding dispatch calls a registry *R*.

Template Parameters

<i>R</i>	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.
----------	---

Definition at line 139 of file [ipc_server_loop](#).

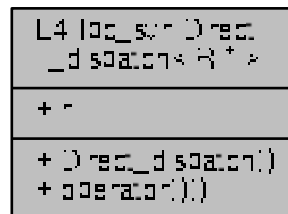
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

14.132 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference

Direct disptach helper, for forwarding dispatch calls via a pointer to a registry *R*.

Collaboration diagram for L4::lpc_svr::Direct_dispatch< R * >:



Public Member Functions

- [Direct_dispatch](#) (*R *r*)
Make a direct dispatcher.
- [l4_msgtag_t operator\(\)](#) (*l4_msgtag_t tag*, *l4_umword_t obj*, *l4_utcb_t *utcb*)
call operator forwarding to r->dispatch()

Data Fields

- *R *r*
stores a pointer to the registry object

14.132.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R * >
```

Direct disptach helper, for forwarding dispatch calls via a pointer to a registry *R*.

Template Parameters

<i>R</i>	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.
----------	---

Definition at line 160 of file [ipc_server_loop](#).

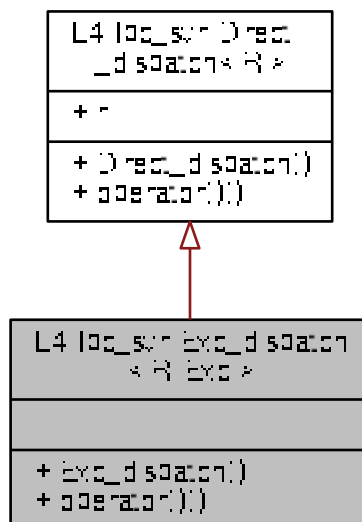
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

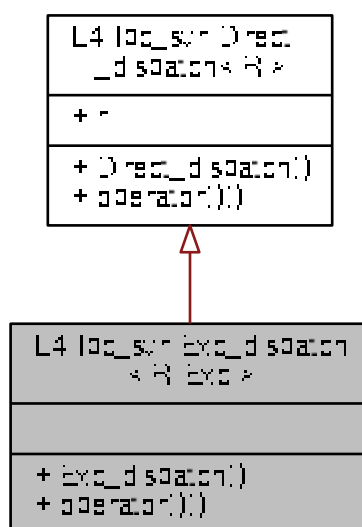
14.133 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference

Dispatch helper wrapping try {} catch {} around the dispatch call.

Inheritance diagram for L4::lpc_svr::Exc_dispatch< R, Exc >:



Collaboration diagram for L4::lpc_svr::Exc_dispatch< R, Exc >:



Public Member Functions

- [Exc_dispatch](#) (R r)
Make an exception handling dispatcher.
- [l4_msgtag_t operator\(\)](#) (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
Dispatch the call via [Direct_dispatch<R>\(\)](#) and handle exceptions.

14.133.1 Detailed Description

```
template<typename R, typename Exc>
struct L4::lpc_svr::Exc_dispatch< R, Exc >
```

Dispatch helper wrapping try {} catch {} around the dispatch call.

Template Parameters

<i>R</i>	Data type of the registry used for dispatching to objects.
<i>Exc</i>	Data type of the exceptions that shall be caught. This data type must provide a member <code>err_no()</code> that returns the negative integer (int) error code for the exception.

This dispatcher wraps `Direct_dispatch<R>` with a try-catch (`Exc`).

Definition at line 184 of file [ipc_server_loop](#).

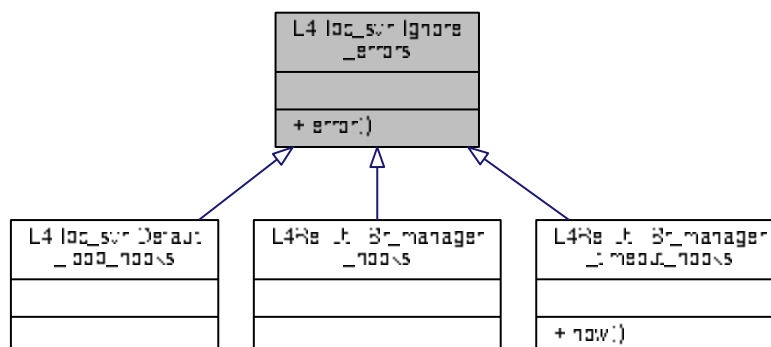
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

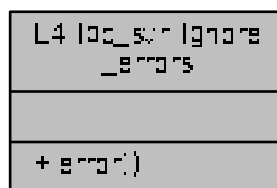
14.134 L4::lpc_svr::Ignore_errors Struct Reference

Mix in for LOOP_HOOKS to ignore IPC errors.

Inheritance diagram for L4::lpc_svr::Ignore_errors:



Collaboration diagram for L4::lpc_svr::ignore_errors:



14.134.1 Detailed Description

Mix in for LOOP_HOOKS to ignore IPC errors.

Definition at line 61 of file [ipc_server_loop](#).

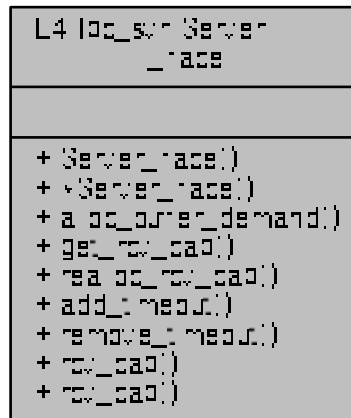
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

14.135 L4::lpc_svr::Server_iface Class Reference

Interface for server-loop related functions.

Collaboration diagram for L4::lpc_svr::Server_iface:



Public Types

- typedef [L4::Type_info::Demand](#) Demand
Data type expressing server-side demand for receive buffers.

Public Member Functions

- [Server_iface](#) ()
Make a server interface.
- virtual int [alloc_buffer_demand](#) ([Demand](#) const &demand)=0
Tells the server to allocate buffers for the given demand.
- virtual [L4::Cap](#)< void > [get_rcv_cap](#) (int index) const =0
Get capability slot allocated to the given receive buffer.
- virtual int [realloc_rcv_cap](#) (int index)=0
Allocate a new capability for the given receive buffer.
- virtual int [add_timeout](#) ([Timeout](#) *timeout, [l4_kernel_clock_t](#) time)=0
Add a timeout to the server internal timeout queue.
- virtual int [remove_timeout](#) ([Timeout](#) *timeout)=0
Remove the given timeout from the timer queue.
- template<typename T >
[L4::Cap](#)< T > [rcv_cap](#) (int index) const
Get given receive buffer as typed capability.
- [L4::Cap](#)< void > [rcv_cap](#) (int index) const
Get receive cap with the given index as generic (void) type.

14.135.1 Detailed Description

Interface for server-loop related functions.

This interface provides access to high-level server-loop related functions, such as management of receive buffers and timeouts.

Definition at line 47 of file [ipc_epiface](#).

14.135.2 Member Function Documentation

14.135.2.1 add_timeout()

```
virtual int L4::Ipc_svr::Server_iface::add_timeout (
    Timeout * timeout,
    l4_kernel_clock_t time ) [pure virtual]
```

Add a timeout to the server internal timeout queue.

Parameters

<i>timeout</i>	The timeout object to register.
<i>time</i>	The time (absolute) at which the timeout shall expire.

Precondition

timeout must not be in any queue.

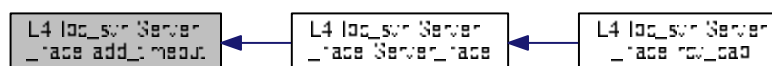
Returns

0 on success, 1 if timeout is already expired, < 0 on error.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), [L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >](#), [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >](#), and [L4Re::Util::Br_manager](#).

Referenced by [Server_iface\(\)](#).

Here is the caller graph for this function:



14.135.2.2 `alloc_buffer_demand()`

```
virtual int L4::Ipc_svr::Server_iface::alloc_buffer_demand (
    Demand const & demand ) [pure virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

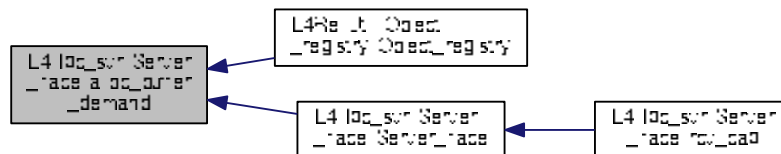
<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see Demand.
---------------	---

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implemented in [L4::Ipc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

Referenced by [L4Re::Util::Object_registry::Object_registry\(\)](#), and [Server_iface\(\)](#).

Here is the caller graph for this function:



14.135.2.3 `get_rcv_cap()`

```
virtual L4::Cap<void> L4::Ipc_svr::Server_iface::get_rcv_cap (
    int index ) const [pure virtual]
```

Get capability slot allocated to the given receive buffer.

Parameters

<i>index</i>	The receive buffer index of the expected capability argument ($0 \leq \text{index} < \text{caps}$ registered with alloc_buffer_demand()).
--------------	---

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

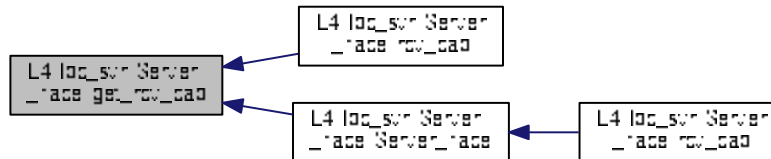
Returns

Capability slot currently allocated to the given receive buffer.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

Referenced by [rcv_cap\(\)](#), and [Server_iface\(\)](#).

Here is the caller graph for this function:

**14.135.2.4 rcv_cap()** [1/2]

```
template<typename T >
L4::Cap<T> L4::lpc_svr::Server_iface::rcv_cap (
    int index ) const [inline]
```

Get given receive buffer as typed capability.

See also

[get_rcv_cap\(\)](#)

Parameters

<i>index</i>	The receive buffer index of the expected capability argument. ($0 \leq \text{index} < \text{caps}$ registered with alloc_buffer_demand() .)
--------------	--

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

Returns

Capability slot currently allocated to the given receive buffer.

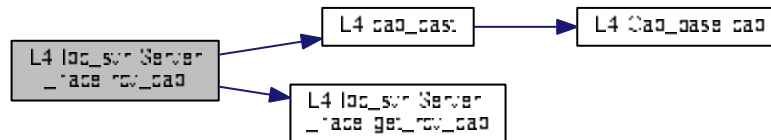
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#) to avoid [L4::cap_cast<>\(\)](#).

Definition at line 122 of file [ipc_epiface](#).

References [L4::cap_cast\(\)](#), and [get_rcv_cap\(\)](#).

Here is the call graph for this function:

**14.135.2.5 rcv_cap()** [2/2]

```
L4::Cap<void> L4::Ipc_svr::Server_iface::rcv_cap (
    int index ) const [inline]
```

Get receive cap with the given index as generic (void) type.

Parameters

<i>index</i>	The index of the cap receive buffer of the expected capability. (0 <= index < caps registered with alloc_buffer_demand() .)
--------------	---

Returns

Capability slot currently allocated to the given capability buffer.

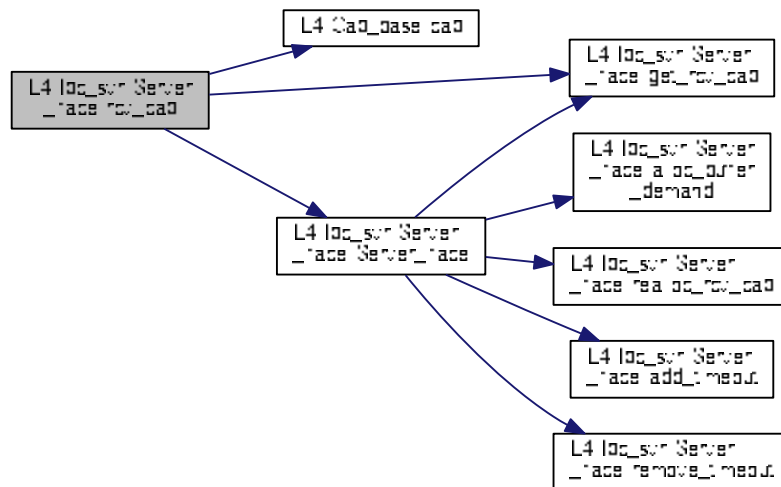
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#).

Definition at line 134 of file [ipc_epiface](#).

References [L4::Cap_base::cap\(\)](#), [get_rcv_cap\(\)](#), [L4_CAP_MASK](#), and [Server_iface\(\)](#).

Here is the call graph for this function:



14.135.2.6 realloc_rcv_cap()

```
virtual int L4::lpc_svr::Server_iface::realloc_rcv_cap (
    int index ) [pure virtual]
```

Allocate a new capability for the given receive buffer.

Parameters

<i>index</i>	The receive buffer index of the expected capability argument ($0 \leq \text{index} < \text{caps}$ registered with alloc_buffer_demand()).
--------------	---

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

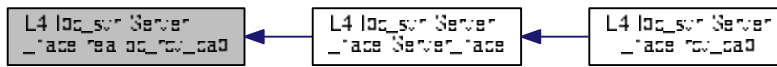
Returns

0 on success, < 0 on error.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

Referenced by [Server_iface\(\)](#).

Here is the caller graph for this function:



14.135.2.7 remove_timeout()

```
virtual int L4::ipc_svr::Server_iface::remove_timeout (
    Timeout * timeout ) [pure virtual]
```

Remove the given timeout from the timer queue.

Parameters

<i>timeout</i>	The timeout object to remove.
----------------	-------------------------------

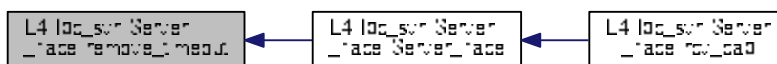
Returns

0 on success, < 0 on error.

Implemented in [L4::ipc_svr::Br_manager_no_buffers](#), [L4::ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >](#), [L4::ipc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >](#), and [L4Re::Util::Br_manager](#).

Referenced by [Server_iface\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

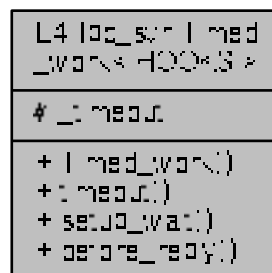
- `l4/sys/cxx/ipc_epiface`

14.136 L4::lpc_svr::Timed_work< HOOKS > Class Template Reference

DEPRECATED.

Inherits HOOKS.

Collaboration diagram for L4::lpc_svr::Timed_work< HOOKS >:



14.136.1 Detailed Description

```
template<typename HOOKS>
class L4::lpc_svr::Timed_work< HOOKS >
```

DEPRECATED.

Deprecated Use [L4::lpc_svr::Timeout_queue_hooks](#)

Definition at line 97 of file [ipc_server_loop](#).

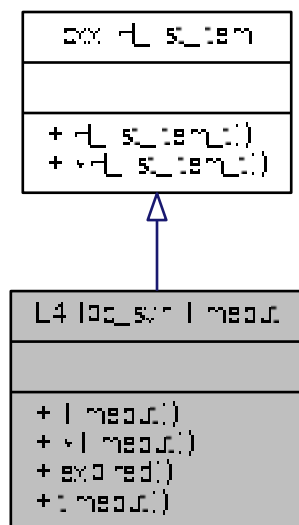
The documentation for this class was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

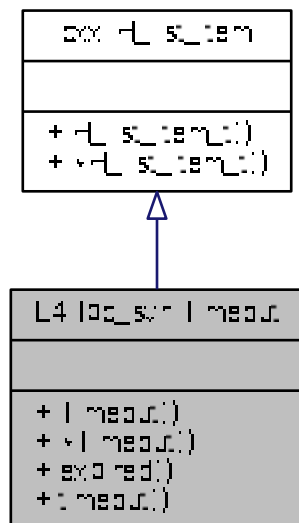
14.137 L4::lpc_svr::Timeout Class Reference

Callback interface for [Timeout_queue](#).

Inheritance diagram for L4::ipc_svr::Timeout:



Collaboration diagram for L4::ipc_svr::Timeout:



Public Member Functions

- [Timeout\(\)](#)

- *Make a timeout.*
virtual [~Timeout](#) ()=0
- *Destroy a timeout.*
virtual void [expired](#) ()=0
callback function to be called when timeout happened
- [l4_kernel_clock_t](#) [timeout](#) () const
return absolute timeout of this callback.

14.137.1 Detailed Description

Callback interface for [Timeout_queue](#).

Definition at line 28 of file [ipc_timeout_queue](#).

14.137.2 Member Function Documentation

14.137.2.1 [expired\(\)](#)

```
virtual void L4::lpc_svr::Timeout::expired ( ) [pure virtual]
```

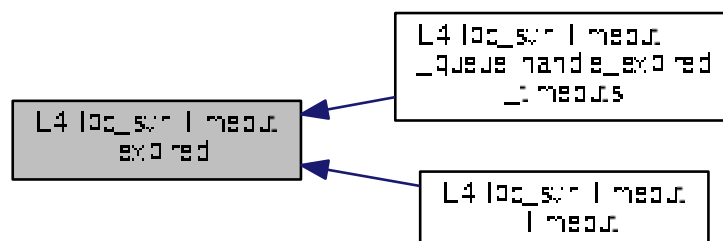
callback function to be called when timeout happened

Note

The timeout object is already dequeued when this function is called, this means the timeout may be safely again within the [expired\(\)](#) function.

Referenced by [L4::lpc_svr::Timeout_queue::handle_expired_timeouts\(\)](#), and [Timeout\(\)](#).

Here is the caller graph for this function:



14.137.2.2 timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout::timeout ( ) const [inline]
```

return absolute timeout of this callback.

Returns

absolute timeout for this instance of the timeout.

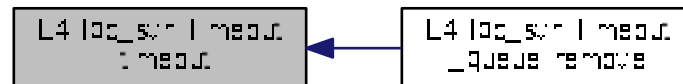
Precondition

The timeout object must have been in a queue before, otherwise the timeout is not set.

Definition at line 51 of file [ipc_timeout_queue](#).

Referenced by [L4::Ipc_svr::Timeout_queue::remove\(\)](#).

Here is the caller graph for this function:



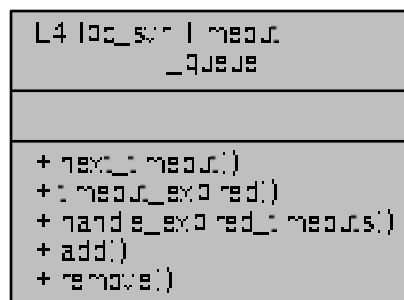
The documentation for this class was generated from the following file:

- `I4/cxx/ipc_timeout_queue`

14.138 L4::Ipc_svr::Timeout_queue Class Reference

Timeout queue to be used in I4re server loop.

Collaboration diagram for `L4::Ipc_svr::Timeout_queue`:



Public Types

- typedef [L4::lpc_svr::Timeout](#) [Timeout](#)
Provide a local definition of [Timeout](#) for backward compat.

Public Member Functions

- [l4_kernel_clock_t](#) [next_timeout](#) () const
Get the time for the next timeout.
- bool [timeout_expired](#) ([l4_kernel_clock_t](#) now) const
Determine if a timeout has happened.
- void [handle_expired_timeouts](#) ([l4_kernel_clock_t](#) now)
run the callbacks of expired timeouts
- void [add](#) ([Timeout](#) *timeout, [l4_kernel_clock_t](#) time)
Add a timeout to the queue.
- void [remove](#) ([Timeout](#) *timeout)
Remove timeout from the queue.

14.138.1 Detailed Description

[Timeout](#) queue to be used in l4re server loop.

Definition at line 64 of file [ipc_timeout_queue](#).

14.138.2 Member Function Documentation

14.138.2.1 add()

```
void L4::Ipc_svr::Timeout_queue::add (
    Timeout * timeout,
    l4\_kernel\_clock\_t time ) [inline]
```

Add a timeout to the queue.

Parameters

<i>timeout</i>	timeout object to add
<i>time</i>	the time when the timeout expires

Precondition

timeout must not be in any queue already

Definition at line 120 of file [ipc_timeout_queue](#).

14.138.2.2 `handle_expired_timeouts()`

```
void L4::Ipc_svr::Timeout_queue::handle_expired_timeouts (
    l4_kernel_clock_t now ) [inline]
```

run the callbacks of expired timeouts

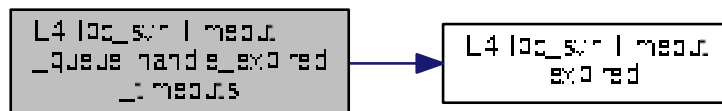
Parameters

<i>now</i>	the current time.
------------	-------------------

Definition at line 100 of file [ipc_timeout_queue](#).

References [L4::Ipc_svr::Timeout::expired\(\)](#).

Here is the call graph for this function:



14.138.2.3 `next_timeout()`

```
l4_kernel_clock_t L4::Ipc_svr::Timeout_queue::next_timeout ( ) const [inline]
```

Get the time for the next timeout.

Returns

the time for the next timeout or 0 if there is none

Definition at line 74 of file [ipc_timeout_queue](#).

14.138.2.4 `remove()`

```
void L4::Ipc_svr::Timeout_queue::remove (
    Timeout * timeout ) [inline]
```

Remove *timeout* from the queue.

Parameters

<i>timeout</i>	timeout to remove from timeout queue
----------------	--------------------------------------

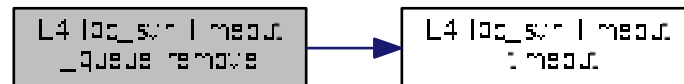
Precondition

timeout must be in this queue

Definition at line 135 of file [ipc_timeout_queue](#).

References [L4::ipc_svr::Timeout::timeout\(\)](#).

Here is the call graph for this function:



14.138.2.5 timeout_expired()

```
bool L4::Ipc_svr::Timeout_queue::timeout_expired (
    l4_kernel_clock_t now ) const [inline]
```

Determine if a timeout has happened.

Parameters

<i>now</i>	The current time.
------------	-------------------

Return values

<i>true</i>	There is at least one expired timeout in the queue. <i>false</i> No expired timeout in the queue.
-------------	---

Definition at line 90 of file [ipc_timeout_queue](#).

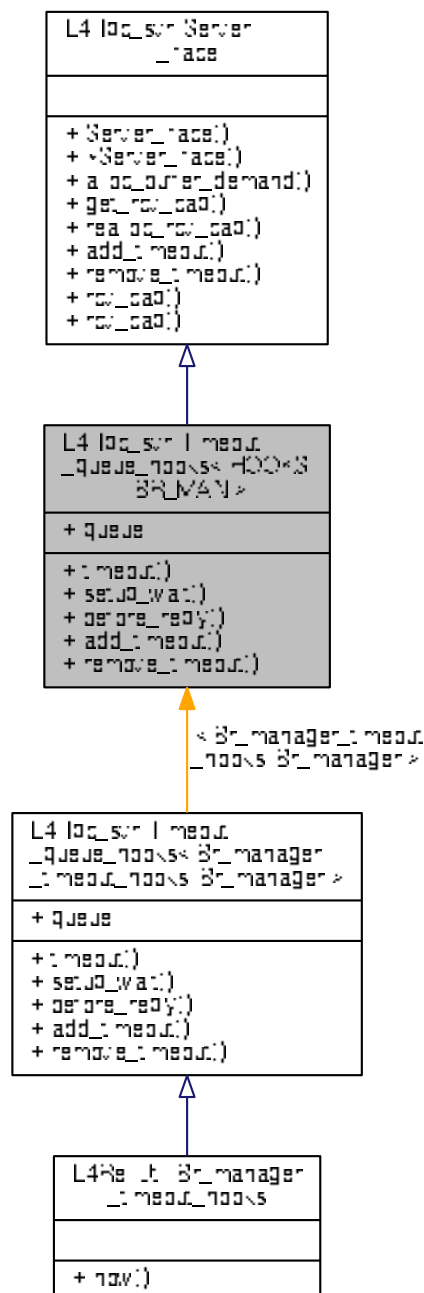
The documentation for this class was generated from the following file:

- `l4/cxx/ipc_timeout_queue`

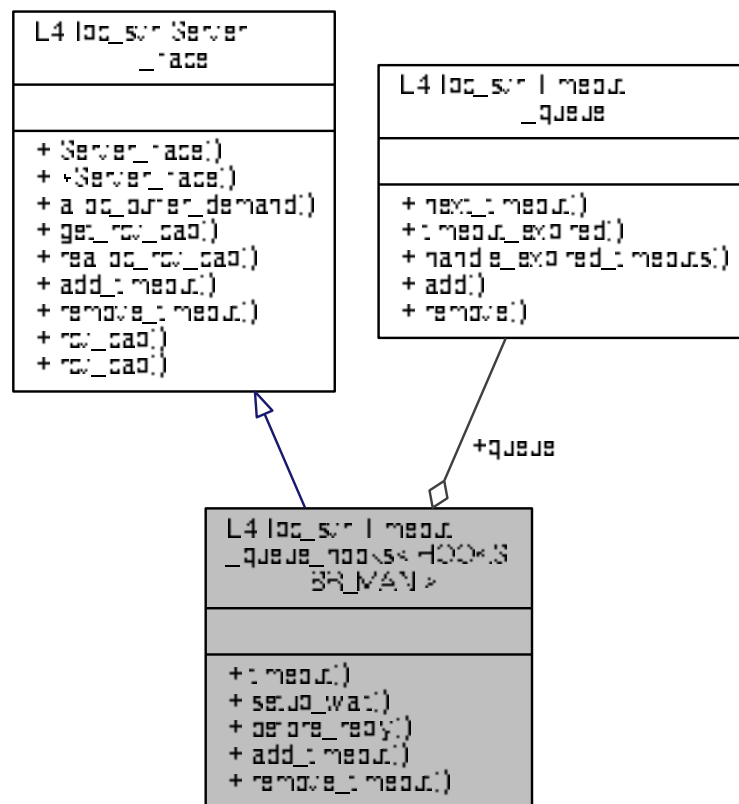
14.139 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference

Loop hooks mixin for integrating a timeout queue into the server loop.

Inheritance diagram for L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >:



Collaboration diagram for L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >:



Public Member Functions

- [l4_timeout_t timeout\(\)](#)
get the time for the next timeout
- void [setup_wait](#)([l4_utcb_t](#) *utcb, [L4::lpc_svr::Reply_mode](#) mode)
setup_wait() for the server loop
- [L4::lpc_svr::Reply_mode before_reply](#)([l4_msgtag_t](#), [l4_utcb_t](#) *)
server loop hook
- int [add_timeout](#)([Timeout](#) *timeout, [l4_kernel_clock_t](#) time)
Add a timeout to the queue for time time.
- int [remove_timeout](#)([Timeout](#) *timeout)
Remove timeout from the queue.

Data Fields

- [Timeout_queue queue](#)
Use this timeout queue.

Additional Inherited Members

14.139.1 Detailed Description

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
class L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >
```

Loop hooks mixin for integrating a timeout queue into the server loop.

Template Parameters

<i>HOOKS</i>	has to inherit from <code>Timeout_queue_hooks<></code> and provide the functions <code>now()</code> that has to return the current time.
<i>BR_MAN</i>	This used as a base class for and provides the API for selecting the buffer register (BR) that is used to store the timeout value. This is usually L4Re::Util::Br_manager or L4::lpc_svr::Br_manager_no_buffers .

Definition at line 159 of file [ipc_timeout_queue](#).

14.139.2 Member Function Documentation

14.139.2.1 add_timeout()

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
int L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::add_timeout (
    Timeout * timeout,
    l4_kernel_clock_t time ) [inline], [virtual]
```

Add a timeout to the queue for time *time*.

Parameters

<i>timeout</i>	The timeout object to add into the queue (must not be in any queue currently).
<i>time</i>	The time when the timeout shall expire.

Precondition

timeout must not be in any queue.

Note

The timeout is automatically dequeued before the [Timeout::expired\(\)](#) function is called

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 211 of file [ipc_timeout_queue](#).

14.139.2.2 remove_timeout()

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
int L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::remove_timeout (
    Timeout * timeout ) [inline], [virtual]
```

Remove timeout from the queue.

Parameters

<i>timeout</i>	The timeout object to be removed from the queue.
----------------	--

Note

This function may be safely called even if the timeout is not currently enqueued.
in [Timeout::expired\(\)](#) the timeout is already dequeued!

Implements [L4::Ipc_svr::Server_iface](#).

Definition at line 224 of file [ipc_timeout_queue](#).

The documentation for this class was generated from the following file:

- [l4/cxx/ipc_timeout_queue](#)

14.140 L4::lrq Class Reference

C++ [lrq](#) interface.



- Generated for L4Re by Doxygen

- `l4_msgtag_t receive (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()`
Unmask and wait for this IRQ.
- `l4_msgtag_t wait (l4_umword_t *label, l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()`
Unmask IRQ and (open) wait for any message.
- `l4_msgtag_t unmask (l4_utcb_t *utcb=l4_utcb()) throw ()`
Unmask IRQ.

Additional Inherited Members

14.140.1 Detailed Description

C++ [Irq](#) interface.

Note

"IRQ" is short for "interrupt request". This is often used interchangeably for "interrupt"

The [Irq](#) class provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs)

[Irq](#) objects can be created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

Include File

```
#include <l4/sys/irq>
```

For the C interface refer to the [IRQs](#) API for an overview.

Examples:

[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 117 of file [irq](#).

14.140.2 Member Function Documentation

14.140.2.1 attach()

```
l4_msgtag_t L4::Irq::attach (
    l4_umword_t label,
    Cap< Thread > const & thread = Cap<Thread>::Invalid,
    l4_utcb_t * utcb = l4_utcb() ) throw () [inline]
```

Attach a thread to this interrupt.

Parameters

<i>label</i>	Identifier of the IRQ (<i>protected label</i> used for messages)
<i>thread</i>	Capability of the thread to attach the IRQ to.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

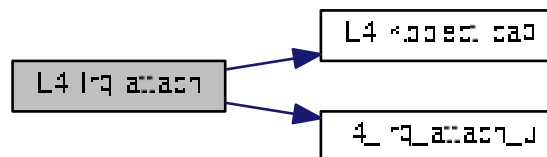
The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Deprecated Use `bind_thread()`.

Definition at line 138 of file `irq`.

References `L4::Kobject::cap()`, and `l4_irq_attach_u()`.

Here is the call graph for this function:



14.140.2.2 detach()

```
l4_msgtag_t L4::Irq::detach (
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

Detach from this interrupt.

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

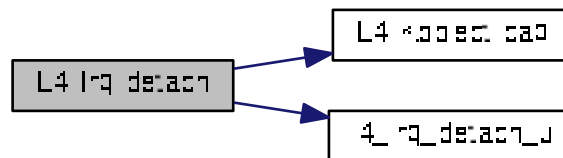
Returns

Syscall return tag

Definition at line 156 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_irq_detach_u\(\)](#).

Here is the call graph for this function:

**14.140.2.3 receive()**

```

l4_msgtag_t L4::Irq::receive (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Unmask and wait for this IRQ.

Parameters

<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

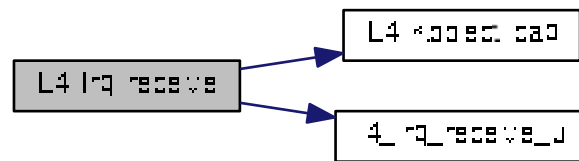
Note

If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 171 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_irq_receive_u\(\)](#).

Here is the call graph for this function:



14.140.2.4 unmask()

```
l4_msgtag_t L4::Irq::unmask (
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

Unmask IRQ.

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

Syscall return tag for a send-only operation, use `l4_ipc_error()` to check for errors (**do not** use `l4_error()`).

Note

This function is a send-only operation, this means there is no return value except for a failed send operation. Use `l4_ipc_error()` to check for errors, **do not** use `l4_error()`, because `l4_error()` will always return an error.

`l4::wait()` and `l4::receive()` operations already include an `unmask()`, do not use an extra `unmask()` in these cases.

Deprecated Use `L4::Irq_eoi::unmask()`

Definition at line 207 of file `irq`.

References `L4_IPC_NEVER`, and `L4::Irq_eoi::unmask()`.

Here is the call graph for this function:



14.140.2.5 wait()

```
l4_msgtag_t L4::Irq::wait (
    l4_umword_t * label,
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Unmask IRQ and (open) wait for any message.

Parameters

<i>label</i>	The <i>protected label</i> shall be received here.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 184 of file [irq](#).

References [L4::Irq_eoi::unmask\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

14.141 L4::Irq_eoi Class Reference

Interface for sending an acknowledge message to an object.

The diagram illustrates the decomposition of the polynomial $L_4^{12}L_5^{10}$ into a product of simpler polynomials. The root node is $L_4^{12}L_5^{10}$, which branches into $L_4^4L_5^5$ and $L_4^8L_5^5$. Each of these further branches into more nodes, eventually leading to a base of $L_4^1L_5^1$. The nodes contain mathematical expressions for the polynomial decomposition at each step, including terms like $L_4^{12}L_5^{10}$, $L_4^{11}L_5^{10}$, $L_4^{10}L_5^{10}$, etc., and their corresponding coefficients.

L4 irq_esp
+ 07 mas<.)

- `l4_msgtag_t unmask` (unsigned irqnum, `l4_umword_t` *label=0, `l4_timeout_t` to=L4_IPC_NEVER, `l4_utcb_t` *utcb=`l4_utcb`()) throw ()
Acknowledge the given interrupt line.

Interface for sending an acknowledge message to an object.

The object is usually an ICU or an IRQ.

See also

[L4::lcu](#), [L4::lrq](#)

Definition at line 43 of file [irq](#).

14.141.2 Member Function Documentation

14.141.2.1 unmask()

```
l4_msgtag_t L4::Irq_eoi::unmask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Acknowledge the given interrupt line.

Parameters

	<i>irqnum</i>	The interrupt line that shall be acknowledged.
out	<i>label</i>	If NULL this is a send-only unmask, if not NULL then this operation enters an open wait and the <i>protected label</i> shall be received here.
	<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
	<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag.

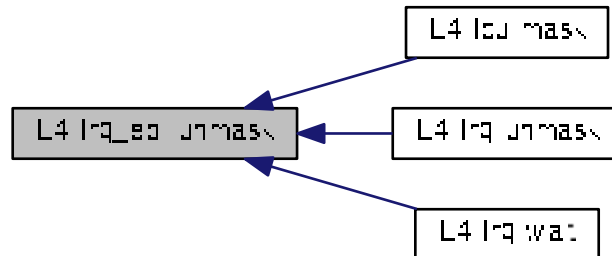
Note

If *label* is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Definition at line 65 of file [irq](#).

Referenced by [L4::lcu::mask\(\)](#), [L4::lrq::unmask\(\)](#), and [L4::lrq::wait\(\)](#).

Here is the caller graph for this function:



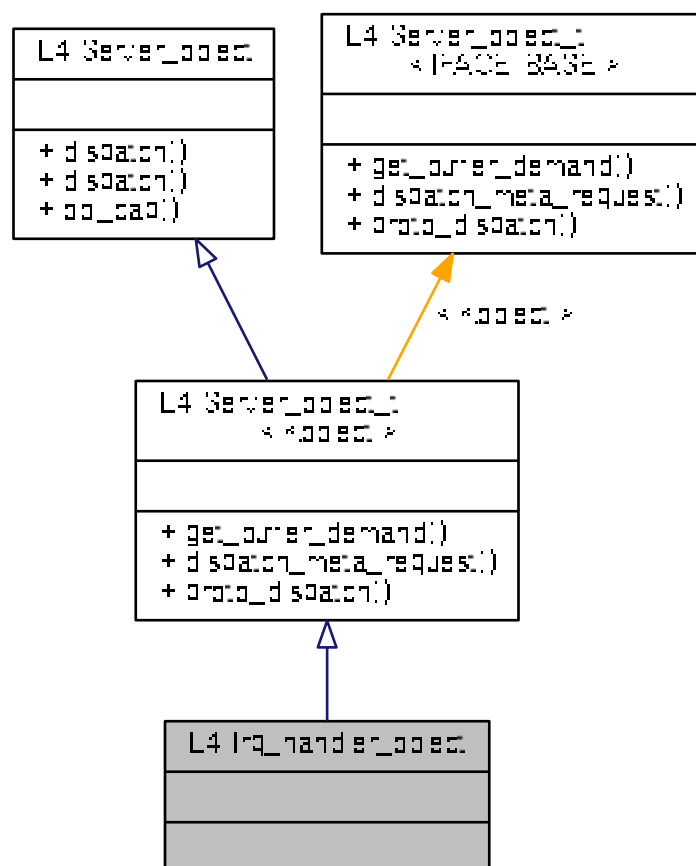
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

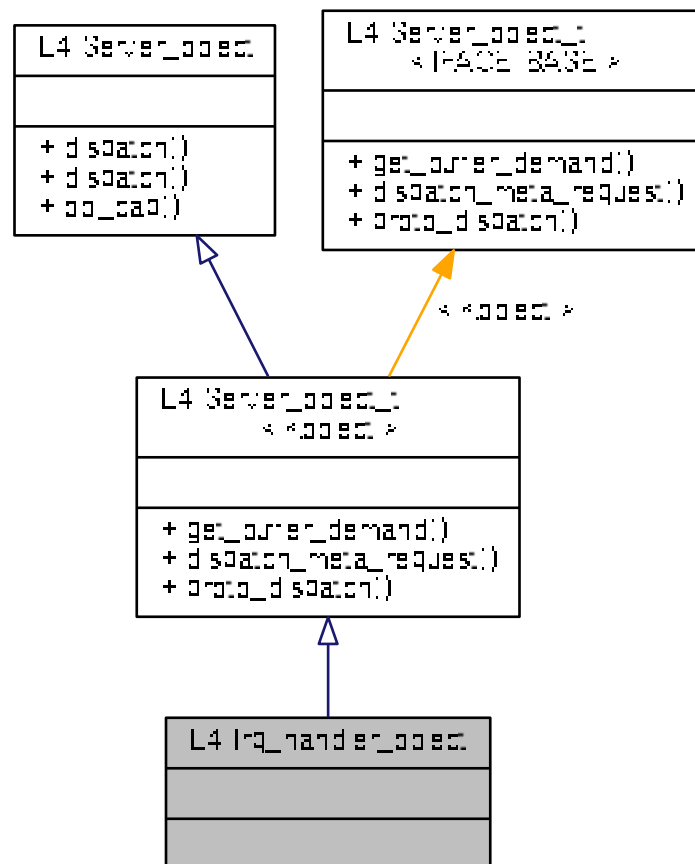
14.142 L4::Irq_handler_object Struct Reference

[Server](#) object base class for handling IRQ messages.

Inheritance diagram for L4::Irq_handler_object:



Collaboration diagram for L4::lrq_handler_object:



Additional Inherited Members

14.142.1 Detailed Description

Server object base class for handling IRQ messages.

This server object base class implements the empty interface ([L4::Kobject](#)). The implementation of [Server_object::dispatch\(\)](#) must return `-L4_ENOREPLY`, because IRQ messages do not handle replies.

Examples:

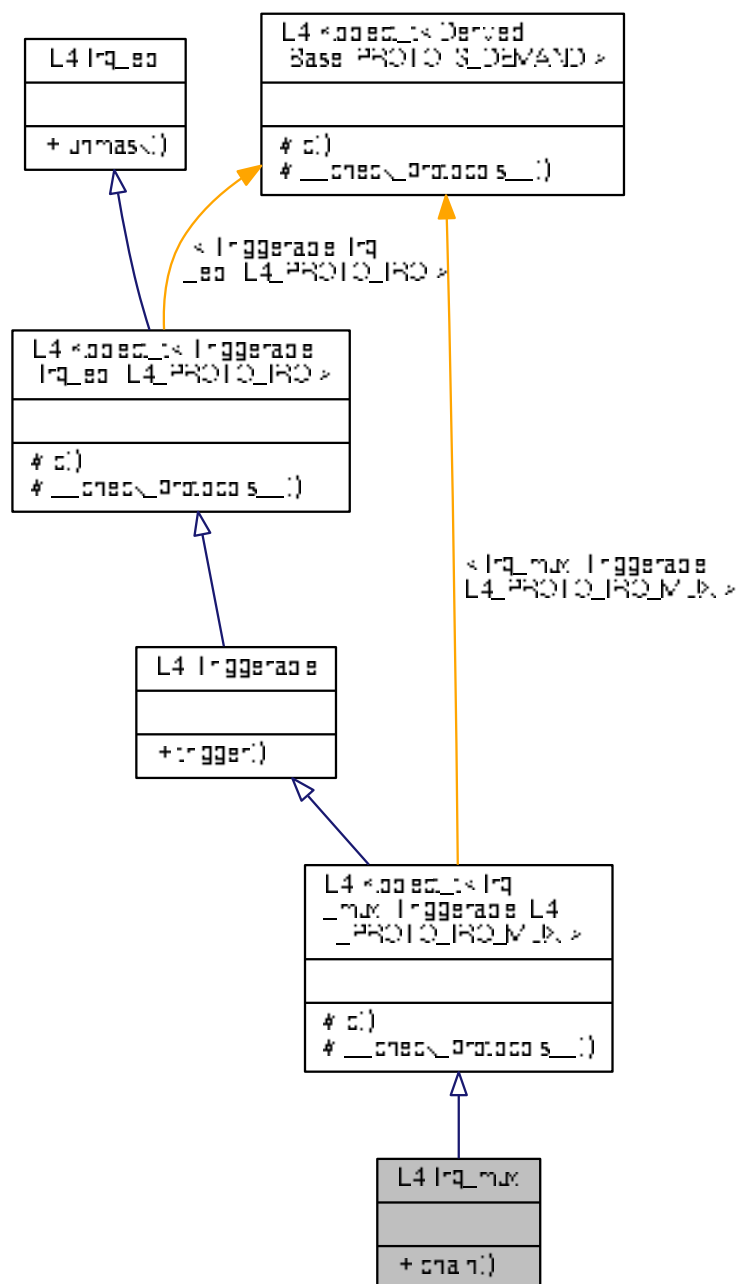
[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 172 of file [ipc_server](#).

The documentation for this struct was generated from the following file:

- [l4/cxx/ipc_server](#)

Collaboration diagram for L4::Irq_mux:



Public Member Functions

- [l4_msgtag_t chain](#) ([Cap](#)< [Triggerable](#) > const &slave, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) throw ()
Attach an IRQ to this multiplexer.

Additional Inherited Members

14.143.1 Detailed Description

IRQ multiplexer for shared IRQs.

This interface allows broadcasting of shared IRQs to multiple triggerables. The IRQ multiplexer is responsible for the correct mask and unmask logic for such shared IRQs.

The semantics are that each of the slave IRQs is triggered whenever the multiplexer IRQ is triggered. As shared IRQs are usually level-triggered, the real IRQ source will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are acknowledged.

Definition at line 224 of file [irq](#).

14.143.2 Member Function Documentation

14.143.2.1 chain()

```
l4_msgtag_t L4::Irq_mux::chain (
    Cap< Triggerable > const & slave,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Attach an IRQ to this multiplexer.

Parameters

<i>slave</i>	The slave that shall be attached to the master.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

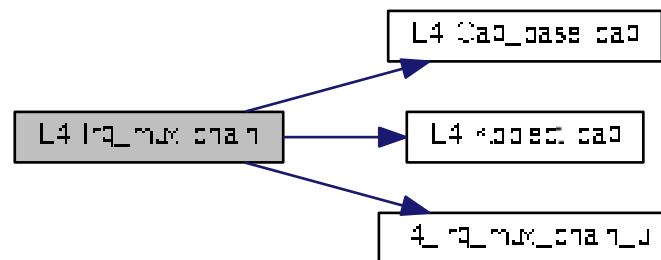
Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer ([Irq_mux](#)) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 239 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_irq_mux_chain_u\(\)](#).

Here is the call graph for this function:



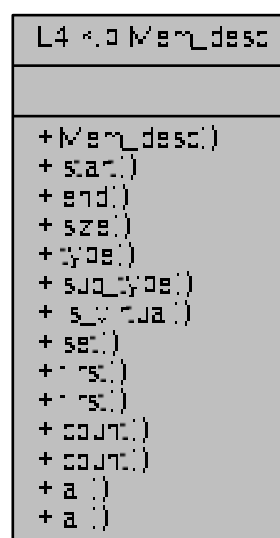
The documentation for this struct was generated from the following file:

- [l4/sys/irq](#)

14.144 L4::Kip::Mem_desc Class Reference

Memory descriptors stored in the kernel interface page.

Collaboration diagram for L4::Kip::Mem_desc:



Public Types

- enum `Mem_type` {
`Undefined` = 0x0, `Conventional` = 0x1, `Reserved` = 0x2, `Dedicated` = 0x3,
`Shared` = 0x4, `Info` = 0xd, `Bootloader` = 0xe, `Arch` = 0xf }
Memory types.
- enum `Info_sub_type` { `Info_acpi_rsdp` = 0 }
Memory sub types for the Mem_type::Info type.

Public Member Functions

- `Mem_desc` (unsigned long `start`, unsigned long `end`, `Mem_type` t, unsigned char st=0, bool virt=false) throw ()
Initialize memory descriptor.
- unsigned long `start` () const throw ()
Return start address of memory descriptor.
- unsigned long `end` () const throw ()
Return end address of memory descriptor.
- unsigned long `size` () const throw ()
Return size of region described by the memory descriptor.
- `Mem_type` type () const throw ()
Return type of the memory descriptor.
- unsigned char `sub_type` () const throw ()
Return sub-type of the memory descriptor.
- unsigned `is_virtual` () const throw ()
Return whether the memory descriptor describes a virtual or physical region.
- void `set` (unsigned long `start`, unsigned long `end`, `Mem_type` t, unsigned char st=0, bool virt=false) throw ()
Set values of a memory descriptor.

Static Public Member Functions

- static `Mem_desc` * `first` (void *kip) throw ()
Get first memory descriptor.
- static unsigned long `count` (void const *kip) throw ()
Return number of memory descriptors stored in the kernel info page.
- static void `count` (void *kip, unsigned count) throw ()
Set number of memory descriptors.
- static `cxx::static_vector`< `Mem_desc` const > `all` (void const *kip)
Return enumerable list of memory descriptors.
- static `cxx::static_vector`< `Mem_desc` > `all` (void *kip)
Return enumerable list of memory descriptors.

14.144.1 Detailed Description

Memory descriptors stored in the kernel interface page.

Include File

```
#include <l4/sys/kip>
```

Definition at line 53 of file `kip`.

14.144.2 Member Enumeration Documentation

14.144.2.1 Info_sub_type

```
enum L4::Kip::Mem_desc::Info_sub_type
```

Memory sub types for the Mem_type::Info type.

Enumerator

Info_acpi_rsdp	Physical address of the ACPI root pointer.
----------------	--

Definition at line 75 of file [kip](#).

14.144.2.2 Mem_type

```
enum L4::Kip::Mem_desc::Mem_type
```

Memory types.

Enumerator

Undefined	Undefined memory.
Conventional	Conventional memory.
Reserved	Reserved region, do not use this memory.
Dedicated	Dedicated.
Shared	Shared.
Info	Info by boot loader.
Bootloader	Memory belongs to the boot loader.
Arch	Architecture specific memory.

Definition at line 59 of file [kip](#).

14.144.3 Constructor & Destructor Documentation

14.144.3.1 Mem_desc()

```
L4::Kip::Mem_desc::Mem_desc (  
    unsigned long start,  
    unsigned long end,
```

```
Mem_type t,
unsigned char st = 0,
bool virt = false ) throw )    [inline]
```

Initialize memory descriptor.

Parameters

<i>start</i>	Start address
<i>end</i>	End address
<i>t</i>	Memory type
<i>st</i>	Memory subtype, defaults to 0
<i>virt</i>	True for virtual memory, false for physical memory, defaults to physical

Definition at line 166 of file [kip](#).

14.144.4 Member Function Documentation

14.144.4.1 all() [1/2]

```
static cxx::static_vector<Mem_desc const> L4::Kip::Mem_desc::all (
    void const * kip )    [inline], [static]
```

Return enumerable list of memory descriptors.

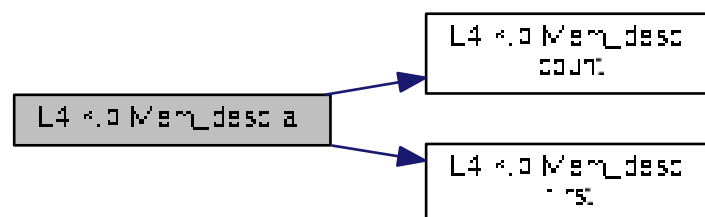
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 139 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



14.144.4.2 all() [2/2]

```
static cxx::static_vector<Mem_desc> L4::Kip::Mem_desc::all (
    void * kip ) [inline], [static]
```

Return enumerable list of memory descriptors.

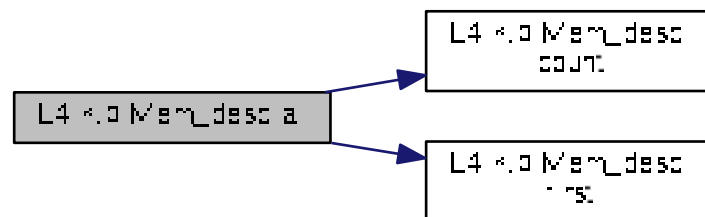
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 150 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



14.144.4.3 count() [1/2]

```
static unsigned long L4::Kip::Mem_desc::count (
    void const * kip ) throw ) [inline], [static]
```

Return number of memory descriptors stored in the kernel info page.

Parameters

<i>kip</i>	Pointer to the kernel info page
------------	---------------------------------

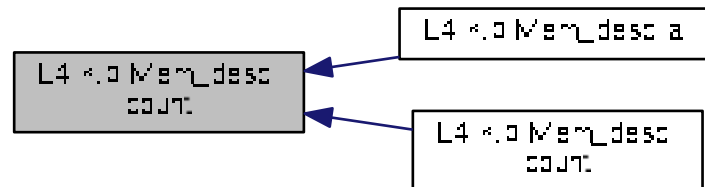
Returns

Number of memory descriptors in the kernel info page.

Definition at line 116 of file [kip](#).

Referenced by [all\(\)](#), and [count\(\)](#).

Here is the caller graph for this function:



14.144.4.4 `count()` [2/2]

```
static void L4::Kip::Mem_desc::count (
    void * kip,
    unsigned count ) throw ()    [inline], [static]
```

Set number of memory descriptors.

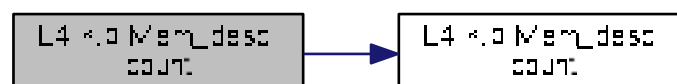
Parameters

<i>kip</i>	Pointer to the kernel info page
<i>count</i>	Number of memory descriptors

Definition at line 128 of file [kip](#).

References [count\(\)](#).

Here is the call graph for this function:



14.144.4.5 end()

```
unsigned long L4::Kip::Mem_desc::end ( ) const throw ( )    [inline]
```

Return end address of memory descriptor.

Returns

End address of memory descriptor

Definition at line 184 of file [kip](#).

Referenced by [size\(\)](#).

Here is the caller graph for this function:



14.144.4.6 first()

```
static Mem_desc* L4::Kip::Mem_desc::first (
    void * kip ) throw ( )    [inline], [static]
```

Get first memory descriptor.

Parameters

<i>kip</i>	Pointer to the kernel info page
------------	---------------------------------

Returns

First memory descriptor stored in the kernel info page

Definition at line 97 of file [kip](#).

Referenced by [all\(\)](#).

Here is the caller graph for this function:



14.144.4.7 `is_virtual()`

```
unsigned L4::Kip::Mem_desc::is_virtual ( ) const throw ( ) [inline]
```

Return whether the memory descriptor describes a virtual or physical region.

Returns

True for virtual region, false for physical region.

Definition at line 213 of file [kip](#).

14.144.4.8 `set()`

```
void L4::Kip::Mem_desc::set (
    unsigned long start,
    unsigned long end,
    Mem_type t,
    unsigned char st = 0,
    bool virt = false ) throw ( ) [inline]
```

Set values of a memory descriptor.

Parameters

<i>start</i>	Start address
<i>end</i>	End address
<i>t</i>	Memory type
<i>st</i>	Sub-type, defaults to 0
<i>virt</i>	Virtual or physical memory region, defaults to physical

Definition at line 224 of file [kip](#).

14.144.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size ( ) const throw ( ) [inline]
```

Return size of region described by the memory descriptor.

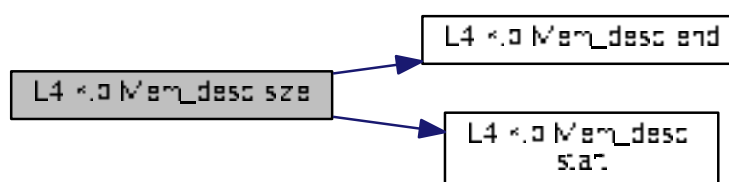
Returns

Size of the region described by the memory descriptor

Definition at line 191 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



14.144.4.10 start()

```
unsigned long L4::Kip::Mem_desc::start ( ) const throw ( ) [inline]
```

Return start address of memory descriptor.

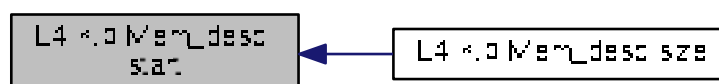
Returns

Start address of memory descriptor

Definition at line 177 of file [kip](#).

Referenced by [size\(\)](#).

Here is the caller graph for this function:



Public Member Functions

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#))
Decrement the in kernel reference counter for the object.

Protected Member Functions

- [l4_cap_idx_t cap](#) () const throw ()
Return capability selector.

14.145.1 Detailed Description

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

Include File

```
#include <l4/sys/capability>
```

This is the base class for all remote objects accessible using RPC. However, subclasses do not directly inherit from [L4::Kobject](#) but *must* use [L4::Kobject_t](#) ([L4::Kobject_0t](#), [L4::Kobject_2t](#), [L4::Kobject_3t](#), or [L4::Kobject_x](#)) for inheritance, otherwise these classes cannot be used as RPC interfaces.

Attention

Objects derived from [Kobject](#) *must* never add any data to those objects. Kobjects can act only as proxy object for encapsulating object invocations.

Definition at line 46 of file [kobject](#).

14.145.2 Member Function Documentation

14.145.2.1 cap()

```
l4\_cap\_idx\_t L4::Kobject::cap ( ) const throw ( )    [inline], [protected]
```

Return capability selector.

Returns

Capability selector.

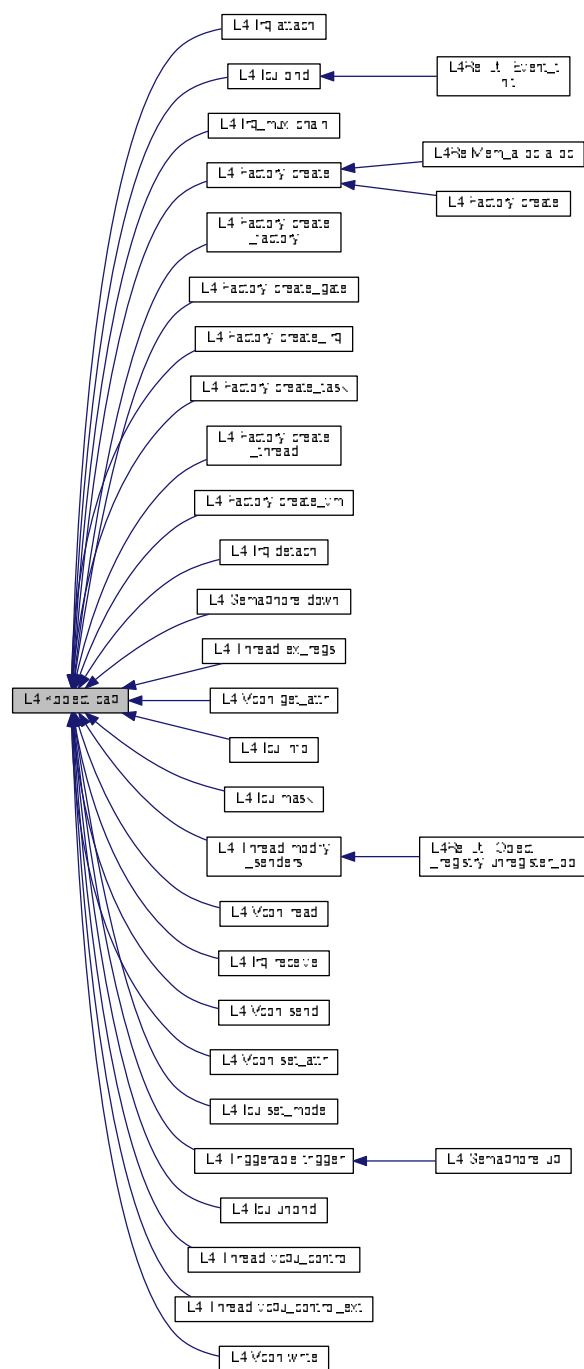
This method is for derived classes to gain access to the actual capability selector.

Definition at line 79 of file [kobject](#).

References [L4_CAP_MASK](#).

Referenced by [L4::Irq::attach\(\)](#), [L4::Icu::bind\(\)](#), [L4::Irq_mux::chain\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create_↵_factory\(\)](#), [L4::Factory::create_gate\(\)](#), [L4::Factory::create_irq\(\)](#), [L4::Factory::create_task\(\)](#), [L4::Factory::create_↵thread\(\)](#), [L4::Factory::create_vm\(\)](#), [L4::Irq::detach\(\)](#), [L4::Semaphore::down\(\)](#), [L4::Thread::ex_regs\(\)](#), [L4::Vcon↵::get_attr\(\)](#), [L4::Icu::info\(\)](#), [L4::Icu::mask\(\)](#), [L4::Thread::modify_senders\(\)](#), [L4::Vcon::read\(\)](#), [L4::Irq::receive\(\)](#), [L4↵::Vcon::send\(\)](#), [L4::Vcon::set_attr\(\)](#), [L4::Icu::set_mode\(\)](#), [L4::Triggerable::trigger\(\)](#), [L4::Icu::unbind\(\)](#), [L4::Thread↵::vcpu_control\(\)](#), [L4::Thread::vcpu_control_ext\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the caller graph for this function:



14.145.2.2 dec_refcnt()

```

14_msgtag_t L4::Kobject::dec_refcnt (
    14_mword_t diff,
    14_utcb_t * utcb = 14_utcb() ) [inline]

```

Decrement the in kernel reference counter for the object.

Parameters

<i>diff</i>	The delta that shall be subtracted from the reference count.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

This function is intended for servers to be able to remove the servers own capability from the counted references. This leads to the semantics that the kernel will delete the object even if the capability of the server is valid. The server can detect the deletion by polling its capabilities or by using the IPC-gate deletion IRQs. And to cleanup if the clients dropped the last reference (capability) to the object.

Definition at line 104 of file [kobject](#).

The documentation for this class was generated from the following file:

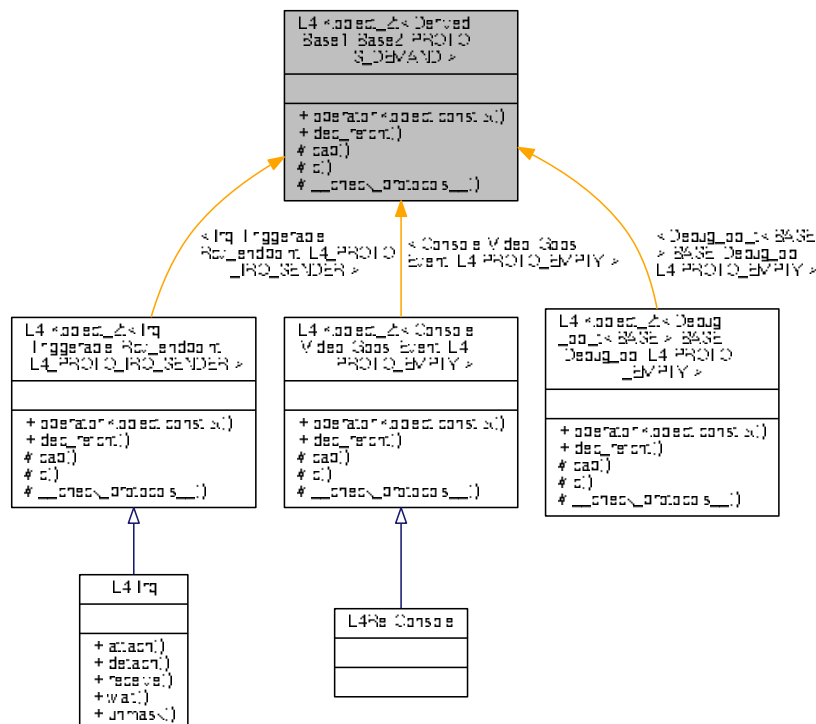
- [l4/sys/kobject](#)

14.146 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference

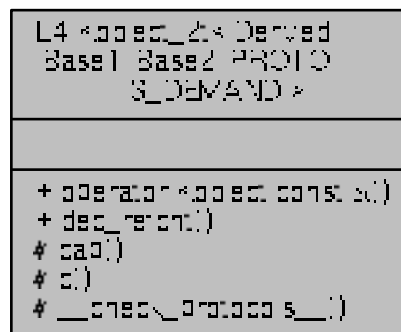
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >:



Collaboration diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >:



Protected Types

- typedef Derived [Class](#)
The target interface type (inheriting from [Kobject_t](#))
- typedef Typeid::Iface< PROTO, Derived > [__Iface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, Typeid::Merge_list< typename Base1::__Iface_list, typename Base2::__Iface_list > > [__Iface_list](#)
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- [L4::Cap< Class > c](#) () const
Get the capability to ourselves.

Static Protected Member Functions

- static void [__check_protocols__](#) ()

14.146.1 Detailed Description

```

template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename S_DEMAND = Type_
info::Demand_t<>>
class L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >

```

Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base1</i>	is the name of the interface's first base class.
<i>Base2</i>	is the name of the interface's second base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand of server-side resources for this interface, usually a L4::Type_info::Demand_t . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface `My_iface` that is derived from [L4::Icu](#) and [L4Re::Dataspace](#).

```
class My_iface : public L4::Kobject_2t<My_iface, L4::Icu, L4Re::Dataspace>
{
    ...
};
```

Definition at line [835](#) of file [__typeinfo.h](#).

14.146.2 Member Typedef Documentation

14.146.2.1 __iface

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND
>::__Iface [protected]
```

The interface description for the derived class.

Definition at line [841](#) of file [__typeinfo.h](#).

14.146.2.2 __iface_list

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<__Iface>, Typeid::Merge_list< typename Base1↵
::__Iface_list, typename Base2::__Iface_list > > L4::Kobject_2t< Derived, Base1, Base2, PRO↵
TO, S_DEMAND >::__Iface_list [protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line [849](#) of file [__typeinfo.h](#).

14.146.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from [Kobject_t](#))

Definition at line 839 of file [__typeinfo.h](#).

14.146.3 Member Function Documentation

14.146.3.1 __check_protocols__()

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::__check_protocols__ ( )
[inline], [static], [protected]
```

Definition at line 852 of file [__typeinfo.h](#).

14.146.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
L4::Cap<Class> L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::c ( ) const [inline],
[protected]
```

Get the capability to ourselves.

Definition at line 871 of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

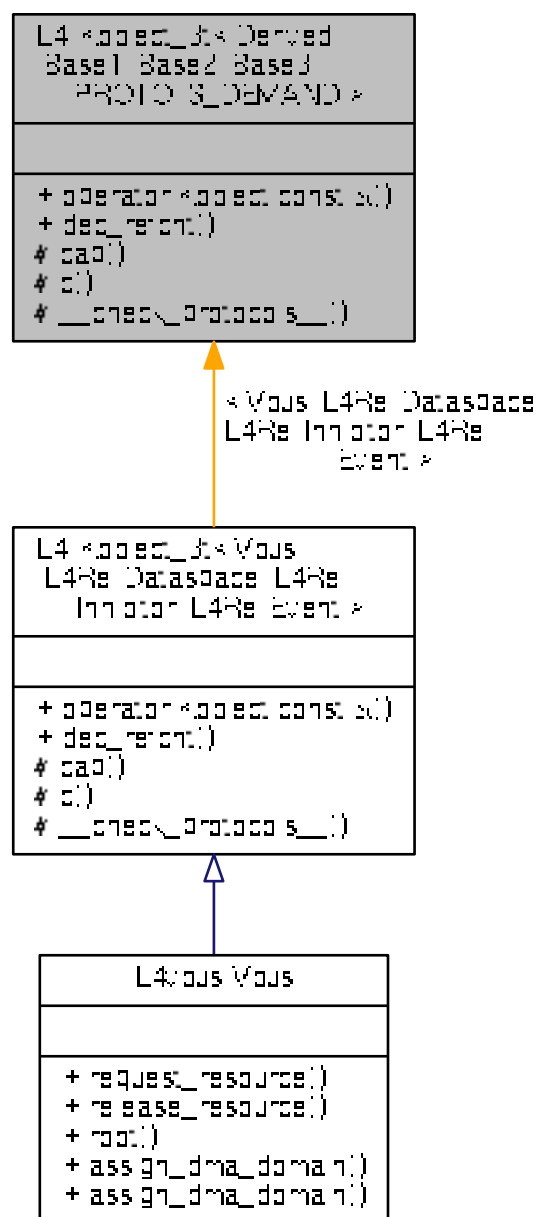
- [l4/sys/__typeinfo.h](#)

14.147 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference

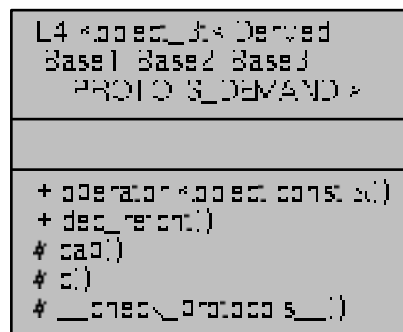
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >:



Collaboration diagram for L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >:



Protected Types

- typedef Derived [Class](#)
The target interface type (inheriting from [Kobject_t](#))
- typedef Typeid::Iface< PROTO, Derived > [__Iface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, Typeid::Merge_list< typename Base1::__Iface_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface_list > > > [__Iface_list](#)
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- [L4::Cap< Class > c](#) () const
Get the capability to ourselves.

Static Protected Member Functions

- static void [__check_protocols__](#) ()

14.147.1 Detailed Description

```

template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
struct L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >

```

Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).

Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base1</i>	is the name of the interface's first base class.
<i>Base2</i>	is the name of the interface's second base class.
<i>Base3</i>	is the name of the interfaces third base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand on server-side resources for this interface, usually a L4::Type_info::Demand_t . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included.

See also

[L4::Kobject_t](#), [L4::Kobject_2t](#), [L4::Kobject_0t](#), [L4::Kobject_x](#)

Definition at line 935 of file [__typeinfo.h](#).

14.147.2 Member Typedef Documentation

14.147.2.1 [__iface](#)

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO,
S_DEMAND >::__iface [protected]
```

The interface description for the derived class.

Definition at line 941 of file [__typeinfo.h](#).

14.147.2.2 [__iface_list](#)

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<\_\_iface>, Typeid::Merge_list< typename Base1↵
::__Iface_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface↵
_list > > > L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__Iface_list
[protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 952 of file [__typeinfo.h](#).

14.147.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from [Kobject_t](#))

Definition at line 939 of file [__typeinfo.h](#).

14.147.3 Member Function Documentation

14.147.3.1 __check_protocols__()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_ANY, typename S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__check_protocols_↵
__ ( ) [inline], [static], [protected]
```

Definition at line 955 of file [__typeinfo.h](#).

14.147.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_ANY, typename S_DEMAND = Type_info::Demand_t<>>
L4::Cap<Class> L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::c ( ) const
[inline], [protected]
```

Get the capability to ourselves.

Definition at line 983 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

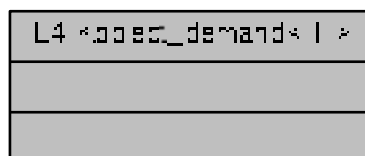
- [l4/sys/__typeinfo.h](#)

14.148 L4::Kobject_demand< T > Struct Template Reference

Get the combined server-side resource requirements for all type T...

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Kobject_demand< T >:



14.148.1 Detailed Description

```
template<typename ... T>
struct L4::Kobject_demand< T >
```

Get the combined server-side resource requirements for all type T...

Template Parameters

T	List of IPC interface types for which the combined server-side resource requirements shall be calculated.
----------	---

Definition at line 1033 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

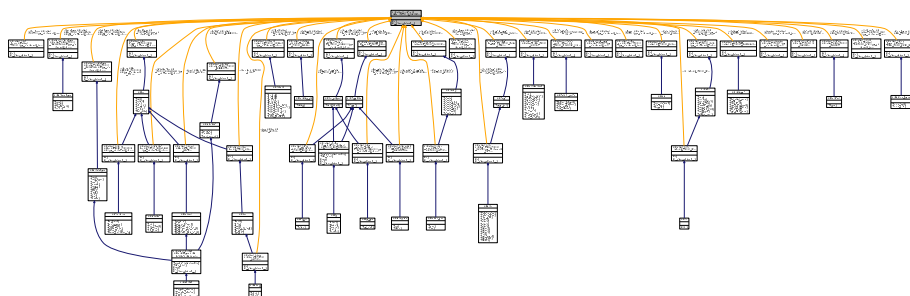
- [l4/sys/__typeinfo.h](#)

14.149 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference

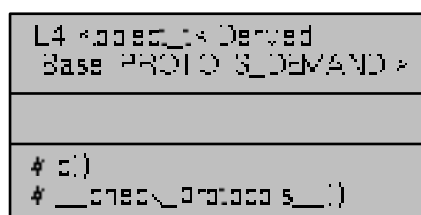
Helper class to create an [L4Re](#) interface class that is derived from a single base class.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >:



Collaboration diagram for L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >:



Protected Types

- typedef Derived [Class](#)
The target interface type (inheriting from [Kobject_t](#))
- typedef Typeid::Iface< PROTO, Derived > [__Iface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename Base::__Iface_list > [__Iface_list](#)
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- [L4::Cap< Class > c \(\)](#) const
Get the capability to ourselves.

Static Protected Member Functions

- static void [__check_protocols__](#) ()
Helper to check for protocol conflicts.

14.149.1 Detailed Description

```
template<typename Derived, typename Base, long PROTO = PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
class L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from a single base class.

Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base</i>	is the name of the interfaces single base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand on server-side resources for this interface, usually a L4::Type_info::Demand_t . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interface <i>Base</i> is automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface *My_iface* that is derived from [L4::Kobject](#).

```
class My_iface : public L4::Kobject_t<My_iface, L4::Kobject>
{
    ...
};
```

Definition at line [759](#) of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

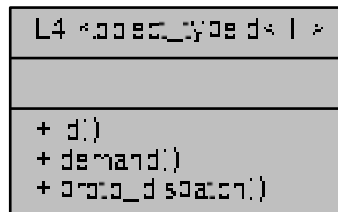
- [l4/sys/__typeinfo.h](#)

14.150 L4::Kobject_typeid< T > Struct Template Reference

[Meta](#) object for handling access to type information of Kobjects.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject_typeid< T >:



Public Types

- typedef T::__Kobject_typeid::Demand [Demand](#)
Data type expressing the static demand of receive buffers in a server.

Static Public Member Functions

- static [Type_info](#) const * [id](#) ()
Get a pointer to the [Kobject](#) type information of T.
- static [Type_info::Demand](#) [demand](#) ()
Get the receive-buffer demand for the server providing the interface T.
- template<typename THIS , typename A1 , typename A2 >
static int [proto_dispatch](#) (THIS *self, long proto, A1 a1, A2 &a2)
Protocol based server-side dispatch function.

14.150.1 Detailed Description

```
template<typename T>
struct L4::Kobject_typeid< T >
```

[Meta](#) object for handling access to type information of Kobjects.

Template Parameters

T	The data type derived from Kobject , usually using Kobject_t .
----------	--

Definition at line 620 of file [__typeinfo.h](#).

14.150.2 Member Typedef Documentation

14.150.2.1 Demand

```
template<typename T>
typedef T::__Kobject_typeid::Demand L4::Kobject_typeid< T >::Demand
```

Data type expressing the static demand of receive buffers in a server.

This information is the combined demand of all base interfaces for T and the buffer demand of T itself. The buffer demand of T is usually specified as the S_DEMAND argument of the [Kobject_t](#) or [Kobject_2t](#) inheritance helpers. S_DEMAND is usually of type [L4::Type_info::Demand_t](#), or [L4::Type_info::Demand_union_t](#).

Definition at line 632 of file [__typeinfo.h](#).

14.150.3 Member Function Documentation

14.150.3.1 demand()

```
template<typename T>
static Type_info::Demand L4::Kobject_typeid< T >::demand ( ) [inline], [static]
```

Get the receive-buffer demand for the server providing the interface T.

Returns

A demand value describing the minimum receive buffers needed for handling server side requests for interface T.

Definition at line 649 of file [__typeinfo.h](#).

14.150.3.2 id()

```
template<typename T>
static Type\_info const* L4::Kobject\_typeid< T >::id ( ) [inline], [static]
```

Get a pointer to the [Kobject](#) type information of T.

Returns

a pointer to the [Kobject](#) typeinfo of T.

Definition at line 640 of file [__typeinfo.h](#).

Referenced by [L4::kobject_typeid\(\)](#).

Here is the caller graph for this function:



14.150.3.3 proto_dispatch()

```
template<typename T>
template<typename THIS , typename A1 , typename A2 >
static int L4::Kobject\_typeid< T >::proto_dispatch (
    THIS * self,
    long proto,
    A1 a1,
    A2 & a2 ) [inline], [static]
```

Protocol based server-side dispatch function.

Template Parameters

$T \leftrightarrow$ <i>HIS</i>	Data type of the server-side object implementing the interface T.
<i>A1</i>	Data type of second argument for <code>p_dispatch()</code>
<i>A2</i>	Data type of third argument for <code>p_dispatch()</code>

Parameters

<i>self</i>	The pointer to the server object
<i>proto</i>	The protocol number used by the caller

Parameters

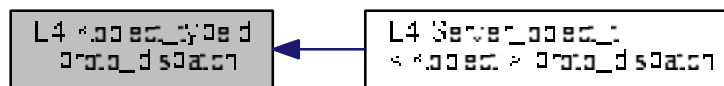
<i>a1</i>	The second argument passed to self->p_dispatch()
<i>a2</i>	The third argument passed to self->p_dispatch()

This function forwards the call to the overloaded p_dispatch() function of self. The data type of the first argument for p_dispatch is determined by the given protocol number.

Definition at line 670 of file [__typeinfo.h](#).

Referenced by [L4::Server_object_t< Kobject >::proto_dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

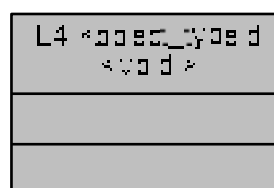
- [l4/sys/__typeinfo.h](#)

14.151 L4::Kobject_typeid< void > Struct Template Reference

Minimalistic ID for void interface.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Kobject_typeid< void >:



14.151.1 Detailed Description

```
template<>
struct L4::Kobject_typeid< void >
```

Minimalistic ID for `void` interface.

Definition at line 677 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

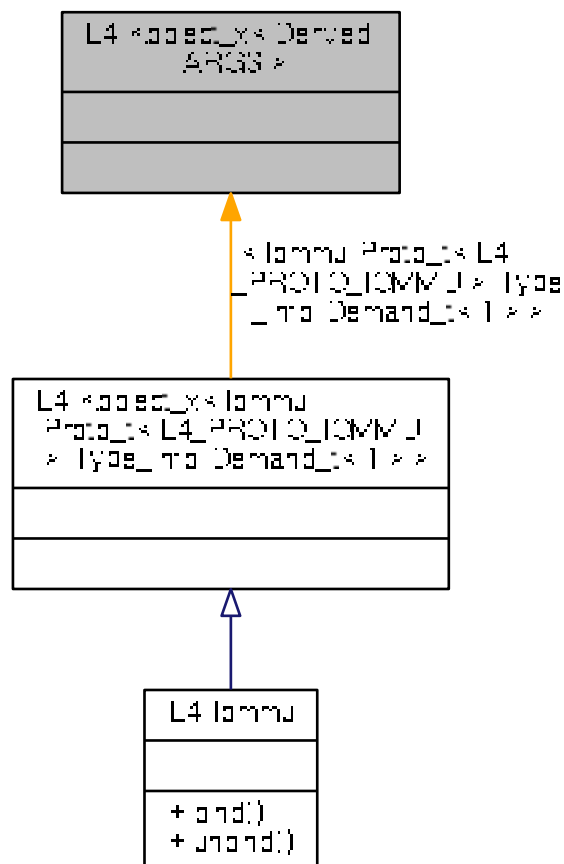
- [l4/sys/__typeinfo.h](#)

14.152 L4::Kobject_x< Derived, ARGS > Struct Template Reference

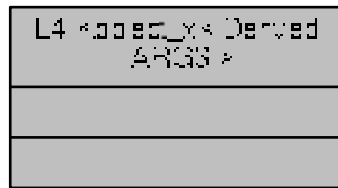
Generic [Kobject](#) inheritance template.

```
#include <l4/sys/capability>
```

Inheritance diagram for `L4::Kobject_x< Derived, ARGS >`:



Collaboration diagram for L4::Kobject_x< Derived, ARGS >:



14.152.1 Detailed Description

```
template<typename Derived, typename ... ARGS>
struct L4::Kobject_x< Derived, ARGS >
```

Generic [Kobject](#) inheritance template.

Template Parameters

<i>Derived</i>	The class name that derives from Kobject_x .
<i>ARGS</i>	An optional protocol number via L4::Proto_t , followed by an optional server-side requirement passed as L4::Type_info::Demand_t , followed by the list of base classes.

Definition at line [1195](#) of file [__typeinfo.h](#).

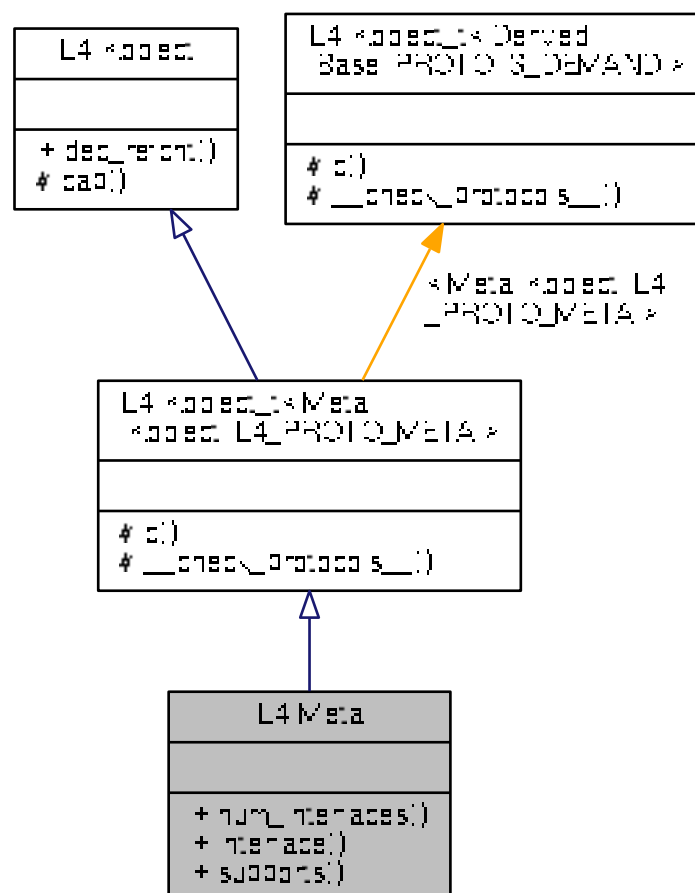
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

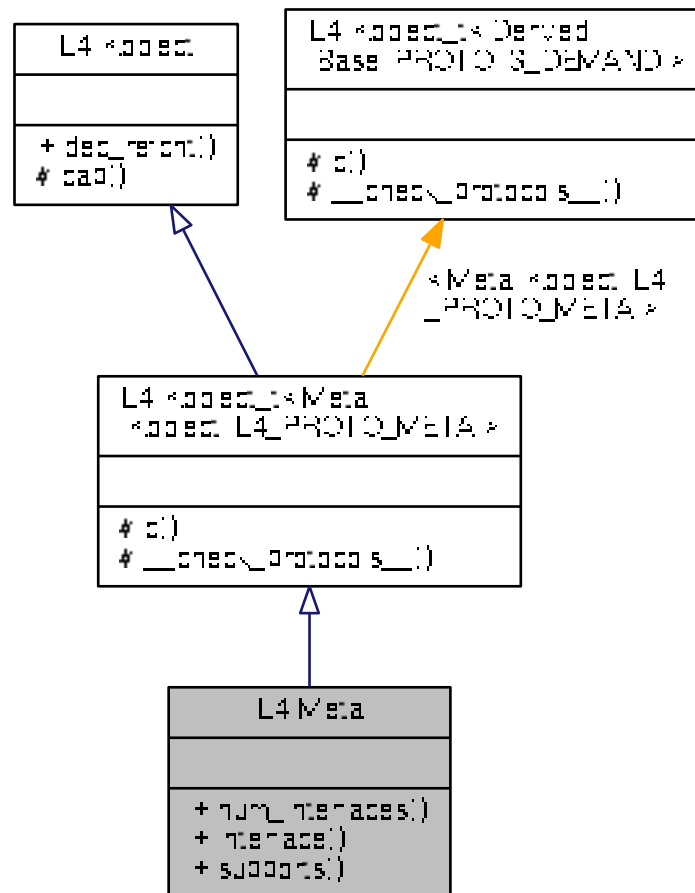
14.153 L4::Meta Class Reference

[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

Inheritance diagram for L4::Meta:



Collaboration diagram for L4::Meta:



Public Member Functions

- [l4_msgtag_t num_interfaces\(\)](#)
Get the number of interfaces implemented by this object.
- [l4_msgtag_t interface\(l4_umword_t idx, long *proto, L4::lpc::String< char > *name\)](#)
*Get the protocol number that must be used for the interface with the number *idx*.*
- [l4_msgtag_t supports\(l4_mword_t protocol\)](#)
Figure out if the object supports the given protocol (number).

Additional Inherited Members

14.153.1 Detailed Description

Meta interface that shall be implemented by each **L4Re** object and gives access to the dynamic type information for **L4Re** objects.

Definition at line 37 of file [meta](#).

14.153.2 Member Function Documentation

14.153.2.1 interface()

```
l4_msgtag_t L4::Meta::interface (
    l4_umword_t idx,
    long * proto,
    L4::Ipc::String< char > * name )
```

Get the protocol number that must be used for the interface with the number `idx`.

Parameters

	<i>idx</i>	The index of the interface to get the protocol number for. <code>idx</code> must be ≥ 0 and $<$ the return value of num_interfaces() .
out	<i>proto</i>	The protocol number for interface <code>idx</code> .
out	<i>name</i>	The protocol name for interface <code>idx</code> .

Return values

l4_msgtag_t::label()	≥ 0 Successful; see <code>proto</code> and <code>name</code> .
l4_msgtag_t::label()	< 0 Error code.

14.153.2.2 num_interfaces()

```
l4_msgtag_t L4::Meta::num_interfaces ( )
```

Get the number of interfaces implemented by this object.

Return values

l4_msgtag_t::label()	≥ 0 The number of supported interfaces.
l4_msgtag_t::label()	< 0 Error code of the occurred error.

14.153.2.3 supports()

```
l4_msgtag_t L4::Meta::supports (
    l4_mword_t protocol )
```

Figure out if the object supports the given protocol (number).

Parameters

<i>protocol</i>	The protocol number to check for.
-----------------	-----------------------------------

Return values

l4_msgtag_t::label()	== 1 protocol is supported.
l4_msgtag_t::label()	== 0 protocol is not supported.

This method is intended to be used for statically assigned protocol numbers.

Referenced by [L4::cap_dynamic_cast\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

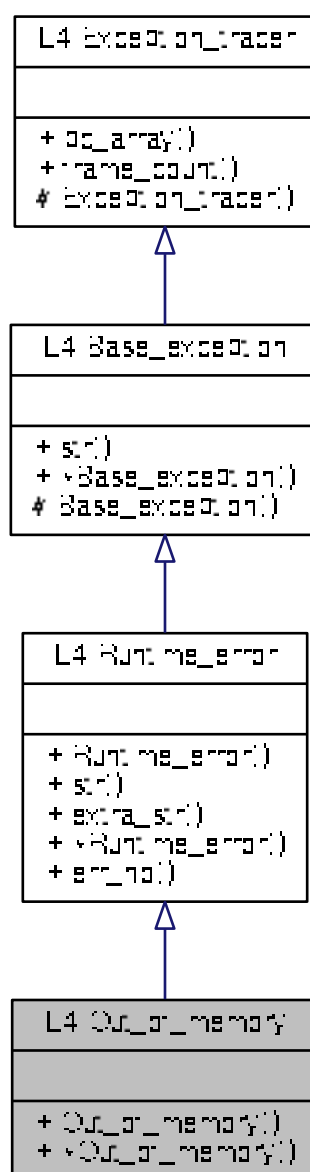
- [l4/sys/meta](#)

14.154 L4::Out_of_memory Class Reference

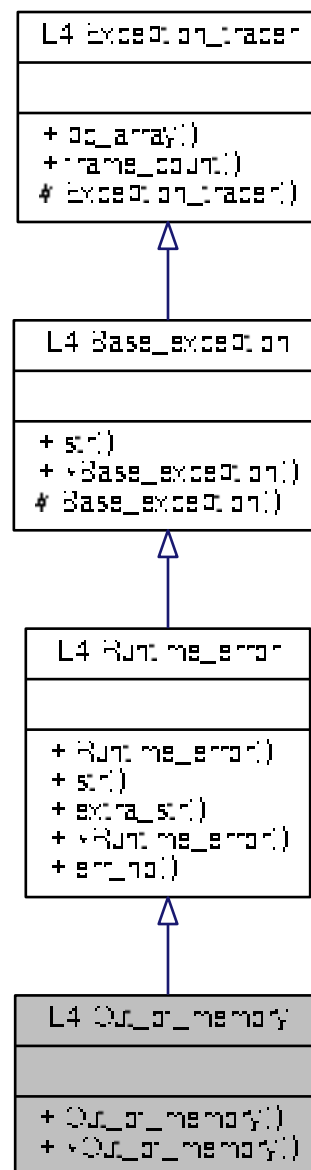
[Exception](#) signalling insufficient memory.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Out_of_memory:



Collaboration diagram for L4::Out_of_memory:



Public Member Functions

- [Out_of_memory](#) (char const *extra="") throw ()

Create an out-of-memory exception.

- [~Out_of_memory](#) () throw ()

Destruction.

Additional Inherited Members

14.154.1 Detailed Description

[Exception](#) signalling insufficient memory.

Definition at line [188](#) of file [exceptions](#).

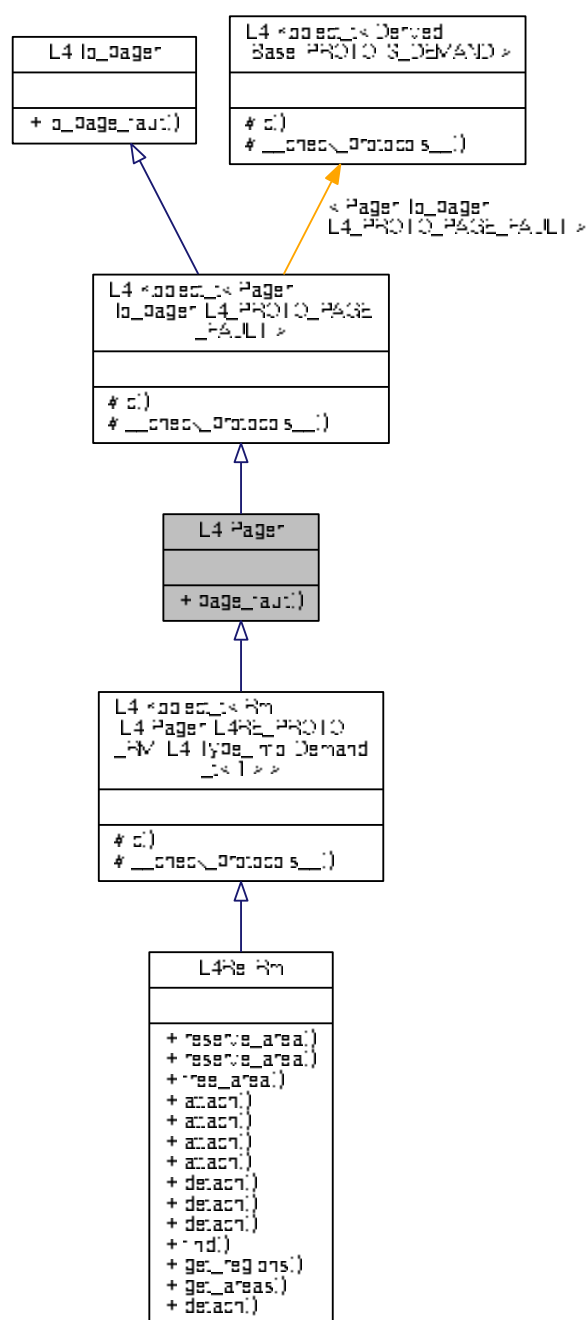
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

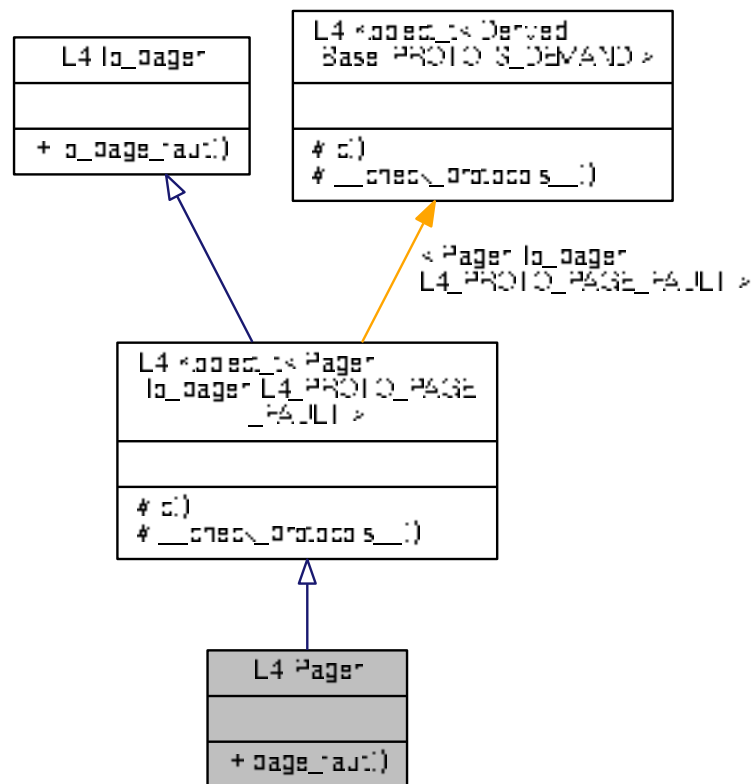
14.155 L4::Pager Class Reference

[Pager](#) interface including the [lo_pager](#) interface.

Inheritance diagram for L4::Pager:



Collaboration diagram for L4::Pager:



Public Member Functions

- `l4_msgtag_t page_fault (l4_umword_t pfa, l4_umword_t pc, L4::lpc::Opt< l4_mword_t > result, L4::lpc::Rcv_fpage rwin, L4::lpc::Opt< L4::lpc::Snd_fpage > fp)`
Page-fault protocol message.

Additional Inherited Members

14.155.1 Detailed Description

`Pager` interface including the `Io_pager` interface.

This class defines the interface for handling page fault IPC. If a thread causes a page fault, the microkernel synthesises a page fault IPC message and sends it to the thread's page fault handler (pager). The pager can then handle the message, for example by establishing a suitable page mapping.

The page fault handler is set with the `L4::Thread::control` interface.

Definition at line 100 of file `pager`.

14.155.2 Member Function Documentation

14.155.2.1 page_fault()

```
l4_msgtag_t L4::Pager::page_fault (
    l4_umword_t pfa,
    l4_umword_t pc,
    L4::Ipc::Opt< l4_mword_t > result,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage > fp )
```

Page-fault protocol message.

Parameters

	<i>pfa</i>	Faulting address including failure reason: bits [0:2]
	<i>pc</i>	Faulting program counter.
out	<i>result</i>	Optional: handling result value.
	<i>rwin</i>	Receive window for a flex-page mapping resolving the page fault
out	<i>fp</i>	Optional: flex-page descriptor to send to the task raising the page fault.

Returns

System call message tag; use [l4_error\(\)](#) to check for errors.

Page-fault messages are usually generated by the kernel and need to be handled by an appropriate handler function that fills in *result* and / or *fp* for the reply.

pfa encoding is as shown:

[63/31 .. 3]	2	1	0
PFA	X	W	T

- **PFA** Bits 63/31..3 of *pfa* are the page fault address bits 63/31 to 3, bits 2..0 are masked.
- **X** Bit 2 of *pfa* is set to 1 for a page fault during instruction fetch.
- **W** Bit 1 of *pfa* is set to 1 for a page fault due to a write operation.
- **T** Bit 0 of *pfa* is set for translation faults (no mapping was present).

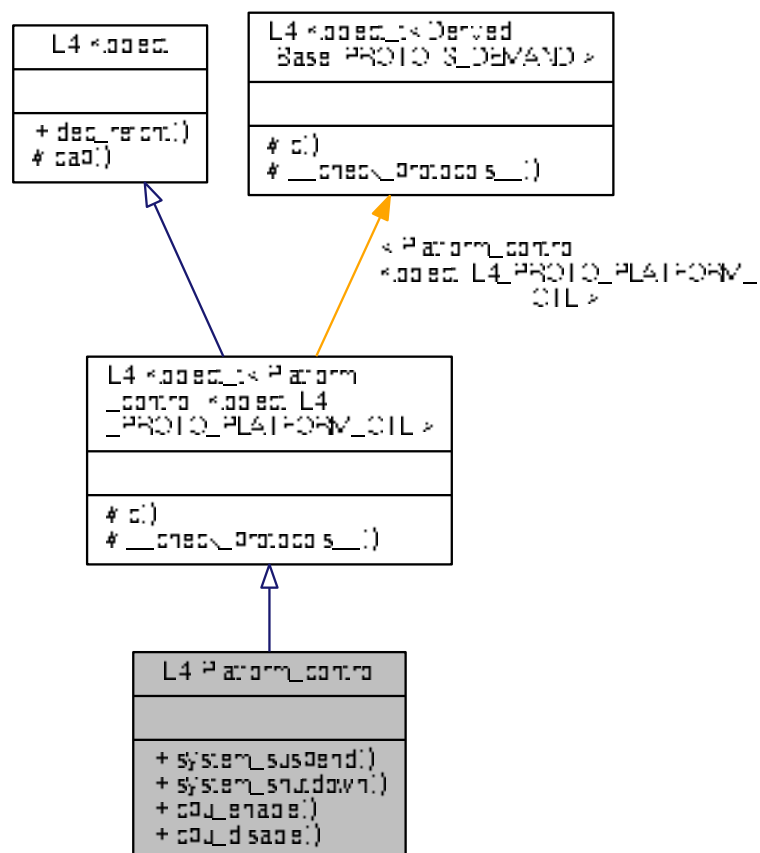
The documentation for this class was generated from the following file:

- [l4/sys/pager](#)

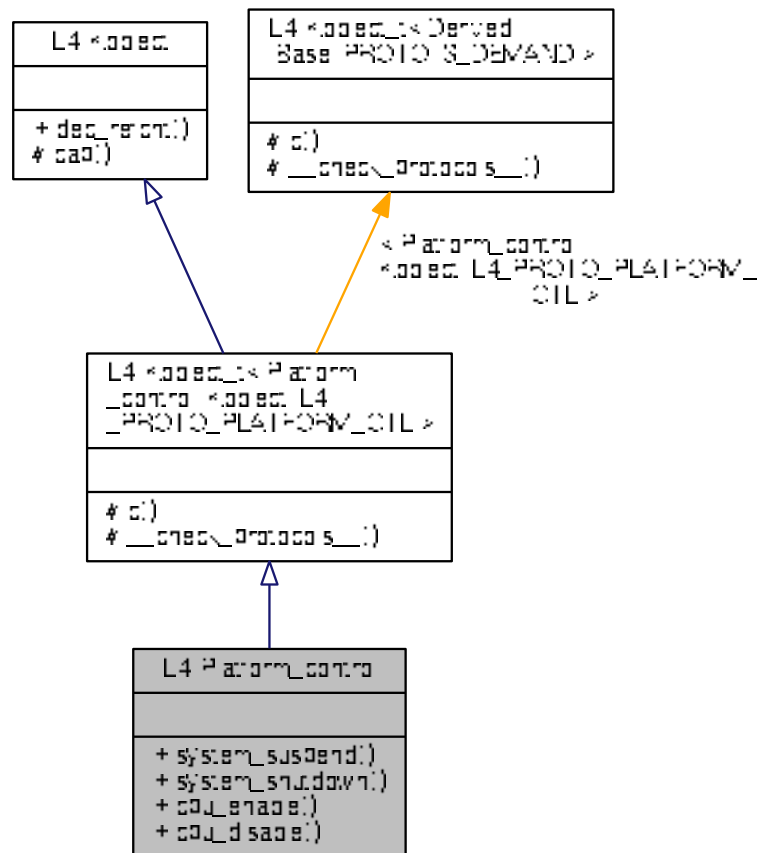
14.156 L4::Platform_control Class Reference

[L4](#) C++ interface for controlling platform-wide properties.

Inheritance diagram for L4::Platform_control:



Collaboration diagram for L4::Platform_control:



Public Types

- enum `Opcode` { `Suspend` = `L4_PLATFORM_CTL_SYS_SUSPEND_OP`, `Shutdown` = `L4_PLATFORM_CTL_SYS_SHUTDOWN_OP`, `Cpu_enable` = `L4_PLATFORM_CTL_CPU_ENABLE_OP`, `Cpu_disable` = `L4_PLATFORM_CTL_CPU_DISABLE_OP` }

Opcodes for platform-control object.

Public Member Functions

- `l4_msgtag_t system_suspend (l4_umword_t extras)`
Enter suspend to RAM.
- `l4_msgtag_t system_shutdown (l4_umword_t reboot)`
Shutdown/Reboot the system.
- `l4_msgtag_t cpu_enable (l4_umword_t phys_id)`
Enable an offline CPU.
- `l4_msgtag_t cpu_disable (l4_umword_t phys_id)`
Disable an online CPU.

Additional Inherited Members

14.156.1 Detailed Description

[L4](#) C++ interface for controlling platform-wide properties.

Add

```
#include <l4/sys/platform_control>
```

to your code to use the platform control functions. The API allows a client to suspend, reboot or shutdown the system.

For the C interface refer to the [Platform Control C API](#).

Definition at line [46](#) of file [platform_control](#).

14.156.2 Member Enumeration Documentation

14.156.2.1 Opcode

```
enum L4::Platform\_control::Opcode
```

Opcodes for platform-control object.

Enumerator

Suspend	Opcode for suspend to RAM.
Shutdown	Opcode for shutdown / reboot.
Cpu_enable	Opcode to enable a CPU.
Cpu_disable	Opcode to disable a CPU.

Definition at line [51](#) of file [platform_control](#).

14.156.3 Member Function Documentation

14.156.3.1 `cpu_disable()`

```
l4\_msgtag\_t L4::Platform\_control::cpu\_disable (  
    l4\_umword\_t phys_id )
```

Disable an online CPU.

Parameters

<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.
----------------	---

Returns

System call message tag

14.156.3.2 `cpu_enable()`

```
l4_msgtag_t L4::Platform_control::cpu_enable (
    l4_umword_t phys_id )
```

Enable an offline CPU.

Parameters

<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.
----------------	--

Returns

System call message tag

14.156.3.3 `system_shutdown()`

```
l4_msgtag_t L4::Platform_control::system_shutdown (
    l4_umword_t reboot )
```

Shutdown/Reboot the system.

Parameters

<i>reboot</i>	1 for reboot, 0 for power off
---------------	-------------------------------

14.156.3.4 `system_suspend()`

```
l4_msgtag_t L4::Platform_control::system_suspend (
    l4_umword_t extras )
```

Enter suspend to RAM.

Parameters

<i>extras</i>	some extra platform-specific information needed to enter suspend to RAM.
---------------	--

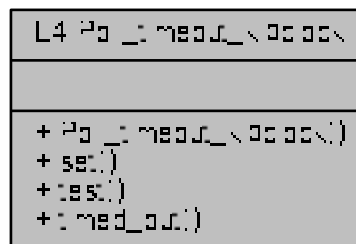
The documentation for this class was generated from the following file:

- [l4/sys/platform_control](#)

14.157 L4::Poll_timeout_kipclock Class Reference

A polling timeout based on the [L4Re](#) clock.

Collaboration diagram for L4::Poll_timeout_kipclock:



Public Member Functions

- [Poll_timeout_kipclock](#) (unsigned poll_time_us)
Initialise relative timeout in microseconds.
- void [set](#) (unsigned poll_time_us)
(Re-)Set relative timeout in microseconds
- bool [test](#) (bool expression=true)
Test whether timeout has expired.
- bool [timed_out](#) () const
Query whether timeout has expired.

14.157.1 Detailed Description

A polling timeout based on the [L4Re](#) clock.

This class allows to conveniently add a timeout to a polling loop.

The original

```
while (device.read(State) & Busy)
;
```

is converted to

```
Poll_timeout_kipclock timeout(10000);
while (timeout.test(device.read(State) & Busy))
;
if (timeout.timed_out())
    printf("ERROR: Device does not respond.\n");
```

Definition at line 37 of file [poll_timeout_kipclock](#).

14.157.2 Constructor & Destructor Documentation

14.157.2.1 Poll_timeout_kipclock()

```
L4::Poll_timeout_kipclock::Poll_timeout_kipclock (
    unsigned poll_time_us ) [inline]
```

Initialise relative timeout in microseconds.

Parameters

<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 44 of file [poll_timeout_kipclock](#).

14.157.3 Member Function Documentation

14.157.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
    unsigned poll_time_us ) [inline]
```

(Re-)Set relative timeout in microseconds

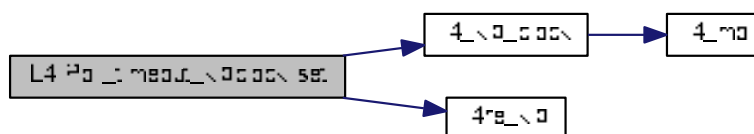
Parameters

<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 53 of file [poll_timeout_kipclock](#).

References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

Here is the call graph for this function:



14.157.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
    bool expression = true ) [inline]
```

Test whether timeout has expired.

Parameters

<i>expression</i>	Optional expression.
-------------------	----------------------

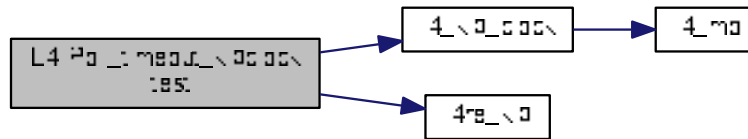
Return values

<i>false</i>	The timeout has expired or the given expression returned false.
<i>true</i>	The timeout has not expired and the optionally given expression returns true.

Definition at line 67 of file [poll_timeout_kipclock](#).

References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

Here is the call graph for this function:



14.157.3.3 timed_out()

```
bool L4::Poll_timeout_kipclock::timed_out ( ) const [inline]
```

Query whether timeout has expired.

Returns

Expiry state of timeout

Definition at line 79 of file [poll_timeout_kipclock](#).

The documentation for this class was generated from the following file:

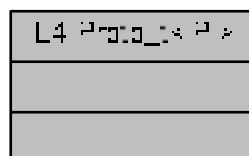
- `l4/re/util/poll_timeout_kipclock`

14.158 L4::Proto_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Proto_t< P >:



14.158.1 Detailed Description

```
template<long P = PROTO_EMPTY>  
struct L4::Proto_t< P >
```

Data type for defining protocol numbers.

Template Parameters

<i>P</i>	The protocol number itself
----------	----------------------------

This type must be used when specifying a protocol number with [Kobject_x](#).

Definition at line 1179 of file [__typeinfo.h](#).

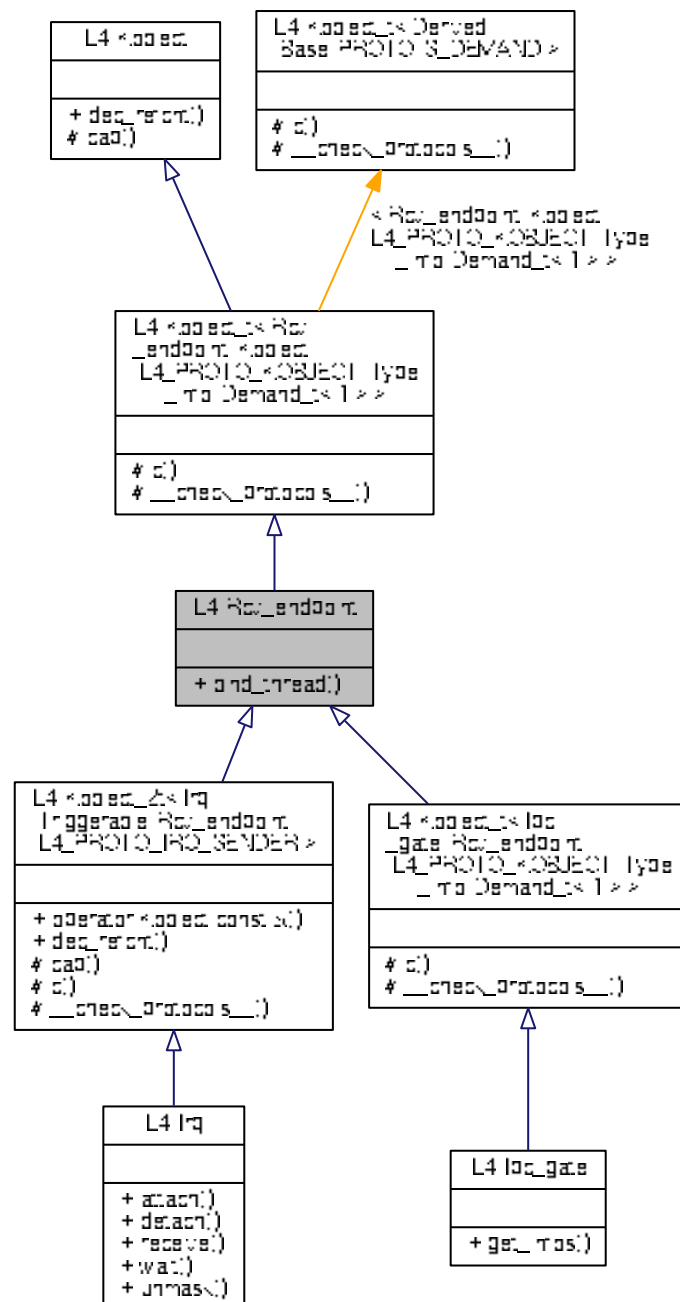
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

14.159 L4::Rcv_endpoint Class Reference

Interface for kernel objects that allow to receive IPC from them.

Inheritance diagram for L4::Rcv_endpoint:



- 14 msqtag t bind thr

- Bind a thread to an IPC receive endpoint.*

1. *Confession* – The confession of sins to God and to the church.

Such an object is for example an `lpc_gate` (with server rights) or an `Irq_sender`. Those objects allow to bind a thread that shall receive IPC from these object via `bind_thread()`.

Generated for L4Re by Doxygen

14.159.2 Member Function Documentation

14.159.2.1 bind_thread()

```
l4_msgtag_t L4::Rcv_endpoint::bind_thread (
    Ipc::Opt< Ipc::Cap< Thread > > t,
    l4_umword_t label )
```

Bind a thread to an IPC receive endpoint.

Parameters

<i>t</i>	Thread object that shall be bound to this receive endpoint.
<i>label</i>	Label to assign to <code>this</code> receive endpoint. The two least significant bits should usually be set to zero.

Returns

Syscall return tag containing one of the following return codes.

Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<code>t</code> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	<code>t</code> is missing L4_CAP_FPAGE_S right.

Referenced by [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the caller graph for this function:



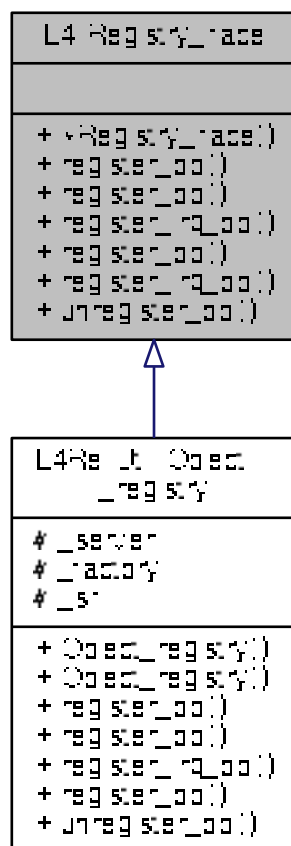
The documentation for this class was generated from the following file:

- [l4/sys/rcv_endpoint](#)

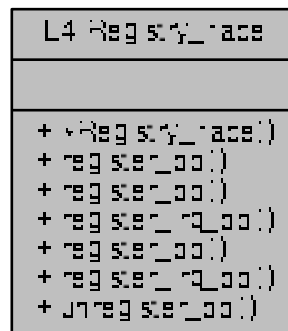
14.160 L4::Registry_iface Class Reference

Abstract interface for object registries.

Inheritance diagram for L4::Registry_iface:



Collaboration diagram for L4::Registry_iface:



Public Member Functions

- virtual [L4::Cap< void > register_obj](#) (L4::Epiface *o, char const *service)=0
Register an L4::Epiface for an IPC gate available in the applications environment under the name `service`.
- virtual [L4::Cap< void > register_obj](#) (L4::Epiface *o)=0
Register o as server-side object for synchronous RPC.
- virtual [L4::Cap< L4::Irq > register_irq_obj](#) (L4::Epiface *o)=0
Register o as server-side object for asynchronous IRQs.
- virtual [L4::Cap< L4::Rcv_endpoint > register_obj](#) (L4::Epiface *o, [L4::Cap< L4::Rcv_endpoint > ep](#))=0
Register o as server-side object for a pre-allocated capability.
- virtual void [unregister_obj](#) (L4::Epiface *o, bool unmap=true)=0
Unregister the given object o from the server.

14.160.1 Detailed Description

Abstract interface for object registries.

An object registry allows to register L4::Epiface objects at a server loop either for synchronous RPC messages or for asynchronous IRQ messages.

Definition at line 305 of file [ipc_epiface](#).

14.160.2 Member Function Documentation

14.160.2.1 register_irq_obj()

```
virtual L4::Cap<L4::Irq> L4::Registry_iface::register_irq_obj (
    L4::Epiface * o ) [pure virtual]
```

Register o as server-side object for asynchronous IRQs.

Parameters

<i>o</i>	Pointer to an Epiface object that shall be registered as server-side object for IRQs.
----------	---

Return values

<i>L4::Cap<L4::Irq></i>	Capability to a new IRQ object on success.
<i>L4::Cap<L4::Irq>::Invalid</i>	The allocation of the IRQ has failed.

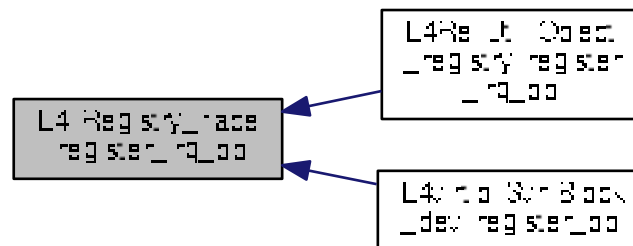
After successful registration `o->obj_cap()` will be the capability of the allocated IRQ object.

The function may allocate a capability slot for the object. In that case `unregister_obj()` is responsible for freeing the slot as well.

Implemented in `L4Re::Util::Object_registry`.

Referenced by `L4Re::Util::Object_registry::register_irq_obj()`, and `L4virtio::Svr::Block_dev<Ds_data>::register_obj()`.

Here is the caller graph for this function:

14.160.2.2 `register_obj()` [1/3]

```
virtual L4::Cap<void> L4::Registry_iface::register_obj (
    L4::Epiface * o,
    char const * service ) [pure virtual]
```

Register an `L4::Epiface` for an IPC gate available in the applications environment under the name `service`.

Parameters

<i>o</i>	Pointer to an Epiface object that shall be registered.
<i>service</i>	Name of the capability that shall be used to connect <code>o</code> to as a server-side object.

Return values

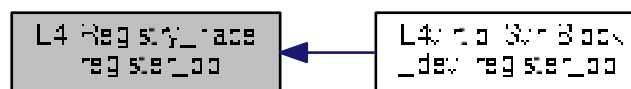
<code>L4::Cap<void></code>	The capability known as <code>service</code> on success.
<code>L4::Cap<void>::Invalid</code>	No capability with the given name found.

After a successful call to this function `o->obj_cap()` is equal to the capability in the environment with the name given by `service`.

Implemented in [L4Re::Util::Object_registry](#).

Referenced by [L4virtio::Svr::Block_dev<Ds_data>::register_obj\(\)](#).

Here is the caller graph for this function:

14.160.2.3 `register_obj()` [2/3]

```
virtual L4::Cap<void> L4::Registry_iface::register_obj (
    L4::Epiface * o ) [pure virtual]
```

Register `o` as server-side object for synchronous RPC.

Parameters

<code>o</code>	Pointer to an Epiface object that shall be registered as server-side object for RPC.
----------------	--

Return values

<code>L4::Cap<void></code>	A valid capability to a new IPC gate.
<code>L4::Cap<void>::Invalid</code>	The allocation of the IPC gate has failed.

After successful registration `o->obj_cap()` will be the capability of the allocated IPC gate.

The function may allocate a capability slot for the object. In that case [unregister_obj\(\)](#) is responsible for freeing the slot as well.

Implemented in [L4Re::Util::Object_registry](#).

14.160.2.4 register_obj() [3/3]

```
virtual L4::Cap<L4::Rcv_endpoint> L4::Registry_iface::register_obj (
    L4::Epiface * o,
    L4::Cap< L4::Rcv_endpoint > ep ) [pure virtual]
```

Register `o` as server-side object for a pre-allocated capability.

Parameters

<i>o</i>	Pointer to an Epiface object that shall be registered as server-side object.
<i>ep</i>	Capability to an already allocated capability where <code>o</code> shall be attached as server-side handler. The capability may point to an IPC gate or an IRQ.

Return values

<i>L4::Cap<L4::Rcv_endpoint></i>	Capability <code>ep</code> on success.
<i>L4::Cap<L4::Rcv_endpoint>::Invalid</i>	The IRQ attach operation has failed.

After successful registration `o->obj_cap()` will be equal to `ep`.

Implemented in [L4Re::Util::Object_registry](#).

14.160.2.5 unregister_obj()

```
virtual void L4::Registry_iface::unregister_obj (
    L4::Epiface * o,
    bool unmap = true ) [pure virtual]
```

Unregister the given object `o` from the server.

Parameters

<i>o</i>	Pointer to the Epiface object that shall be unregistered. The object must have been registered with any of the register methods if Registry_iface .
<i>unmap</i>	If true the capability <code>o->obj_cap()</code> shall be unmapped from the local object space.

The function always unmaps and frees the capability if it was allocated by either [Registry_iface::register_irq_obj\(L4::Epiface *\)](#), or by [Registry_iface::register_obj\(L4::Epiface *\)](#).

Implemented in [L4Re::Util::Object_registry](#).

The documentation for this class was generated from the following file:

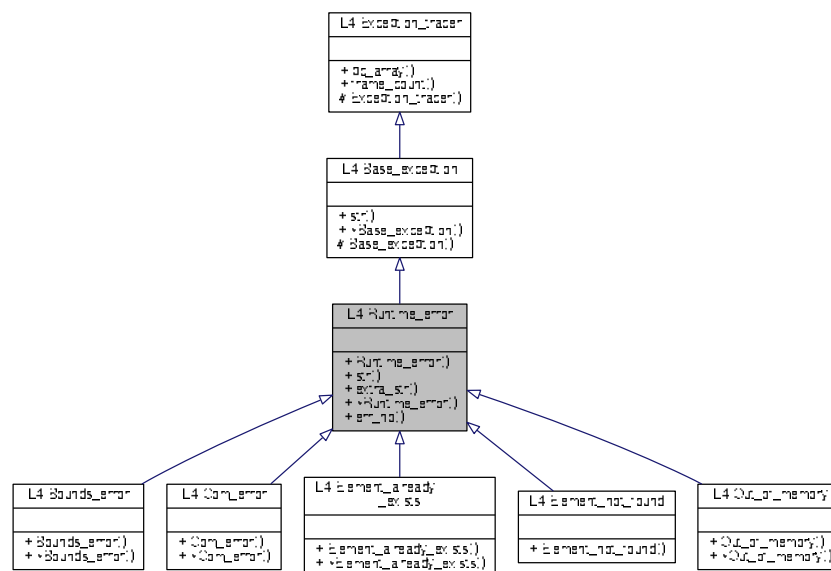
- `l4/sys/cxx/ipc_epiface`

14.161 L4::Runtime_error Class Reference

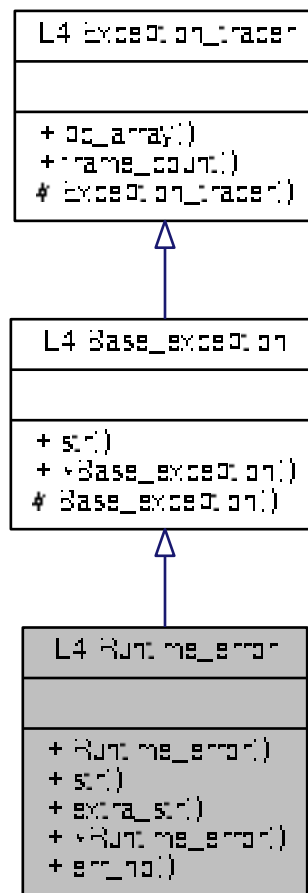
Exception for an abstract runtime error.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Runtime_error:



Collaboration diagram for L4::Runtime_error:



Public Member Functions

- [Runtime_error](#) (long [err_no](#), char const *extra=0) throw ()
Create a new [Runtime_error](#).
- char const * [str](#) () const throw ()
Return a human readable string for the exception.
- char const * [extra_str](#) () const
Get the description text for this runtime error.
- long [err_no](#) () const throw ()
Get the error value for this runtime error.

Additional Inherited Members

14.161.1 Detailed Description

[Exception](#) for an abstract runtime error.

This is the base class for a set of exceptions that cover all errors that have a C error value (see [l4_error_code_t](#)).

Definition at line 139 of file [exceptions](#).

14.161.2 Constructor & Destructor Documentation

14.161.2.1 Runtime_error()

```
L4::Runtime_error::Runtime_error (
    long err_no,
    char const * extra = 0 ) throw ()    [inline], [explicit]
```

Create a new [Runtime_error](#).

Parameters

<i>err_no</i>	Error value for this runtime error.
<i>extra</i>	Description of what was happening while the error occurred.

Definition at line [152](#) of file [exceptions](#).

14.161.3 Member Function Documentation

14.161.3.1 err_no()

```
long L4::Runtime_error::err_no ( ) const throw ()    [inline]
```

Get the error value for this runtime error.

Returns

Error value.

Definition at line [181](#) of file [exceptions](#).

14.161.3.2 extra_str()

```
char const* L4::Runtime_error::extra_str ( ) const    [inline]
```

Get the description text for this runtime error.

Returns

Pointer to the description string.

Definition at line [173](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

C++ interface of the [Scheduler](#) kernel object.

Inheritance diagram for L4::Scheduler:



- `l4_msgtag_t idle_time (l4_sched_cpu_set_t const &cpus, l4_kernel_clock_t *us)`
Query the idle time (in μ s) of a CPU.
- `bool is_online (l4_umword_t cpu, l4_utcb_t *utcb=l4_utcb()) const throw ()`
Query if a CPU is online.

Additional Inherited Members

14.162.1 Detailed Description

C++ interface of the [Scheduler](#) kernel object.

The [Scheduler](#) interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

Include File

```
#include <l4/sys/scheduler>
```

Definition at line 42 of file [scheduler](#).

14.162.2 Member Function Documentation

14.162.2.1 idle_time()

```
l4_msgtag_t L4::Scheduler::idle_time (
    l4_sched_cpu_set_t const & cpus,
    l4_kernel_clock_t * us )
```

Query the idle time (in μ s) of a CPU.

Parameters

	<i>cpus</i>	Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried.
out	<i>us</i>	Idle time of queried CPU in μ s.

Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in cpu set.

This function retrieves the idle time in μ s of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate l one has to retrieve the idle time at the beginning ($i1$) and the end ($i2$) of a known time interval t . The load is then calculated as $l = 1 - (i2 - i1)/t$.

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting `-L4_EINVAL` or calculating an estimated (incorrect) load of 1.

Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

14.162.2.2 info()

```
l4_msgtag_t L4::Scheduler::info (
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_utcb_t * utcb = l4_utcb() ) const throw()    [inline]
```

Get scheduler information.

Parameters

out	<i>cpu_max</i>	Maximum number of CPUs ever available.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs.
	<i>utcb</i>	UTCB pointer of the calling thread. This defaults to the UTCB of the current thread.

Return values

0	Success.
<code>-L4_ERANGE</code>	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 66 of file [scheduler](#).

References [l4_sched_cpu_set_t::gran_offset](#), [L4_INLINE_RPC_OP](#), [L4_SCHEDULER_IDLE_TIME_OP](#), [L4_SCHEDULER_RUN_THREAD_OP](#), [l4_sched_cpu_set_t::map](#), and [cxx::max\(\)](#).

Here is the call graph for this function:



14.162.2.3 `is_online()`

```
bool L4::Scheduler::is_online (
    l4_umword_t cpu,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Query if a CPU is online.

Parameters

<i>cpu</i>	CPU number whose online status should be queried.
<i>utcb</i>	UTCB pointer of the calling thread. Defaults to l4_utcb() .

Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 139 of file [scheduler](#).

14.162.2.4 `run_thread()`

```
l4_msgtag_t L4::Scheduler::run_thread (
    Ipc::Cap< Thread > thread,
    l4_sched_param_t const & sp )
```

Run a thread on a [Scheduler](#).

Parameters

<i>thread</i>	Capability of the thread to run.
<i>sp</i>	Scheduling parameters.

Return values

<i>0</i>	Success.
<i>-L4_EINVAL</i>	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

Note

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different physical CPUs.

- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical CPU.

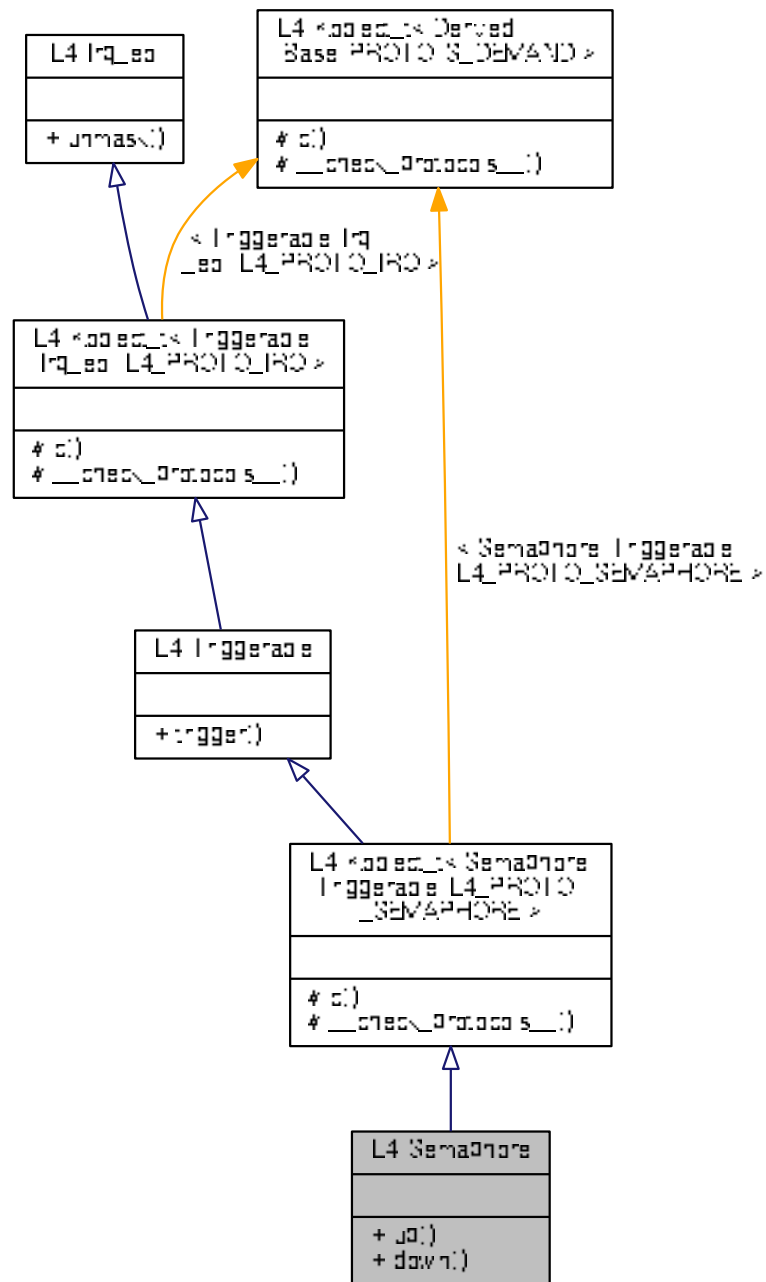
The documentation for this class was generated from the following file:

- [l4/sys/scheduler](#)

14.163 L4::Semaphore Struct Reference

Kernel-provided semaphore object.

Collaboration diagram for L4::Semaphore:



Public Member Functions

- `l4_msgtag_t up (l4_utcb_t *utcb=l4_utcb()) throw ()`
Semaphore up operation (wrapper for `trigger()`).
- `l4_msgtag_t down (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()`
Semaphore down operation.

Additional Inherited Members

14.163.1 Detailed Description

Kernel-provided semaphore object.

This is the interface for kernel-provided semaphore objects. The object provides the classical functions `up()` and `down()` for counting the semaphore and blocking. The semaphore is a [Triggerable](#) with respect to the `up()` function, this means that a semaphore can be bound to an interrupt line at an ICU ([L4::lcu](#)) and incoming interrupts increment the semaphore counter.

The `down()` method decrements the semaphore counter and blocks if the counter is already zero. Blocking on a semaphore may—as all blocking operations—either return successfully, or be aborted due to an expired timeout provided to the `down()` operation, or due to an [L4::Thread::ex_regs\(\)](#) operation with the [L4_THREAD_EX_REGS_CANCEL](#) flag set.

The main reason for using a semaphore instead of an [L4::lrc](#) is to ensure that incoming trigger signals do not interfere with any open-wait operations, as used for example in a server loop.

Definition at line 51 of file [semaphore](#).

14.163.2 Member Function Documentation

14.163.2.1 down()

```
l4_msgtag_t L4::Semaphore::down (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

[Semaphore](#) down operation.

Parameters

<i>timeout</i>	Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

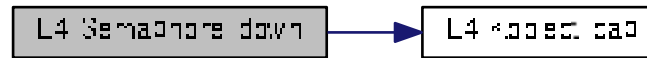
IPC message return tag. Use [l4_ipc_error\(\)](#) to check for a timeout or a cancel condition.

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an `up()` operation.

Definition at line 84 of file [semaphore](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



14.163.2.2 up()

```

l4_msgtag_t L4::Semaphore::up (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

Semaphore up operation (wrapper for [trigger\(\)](#)).

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

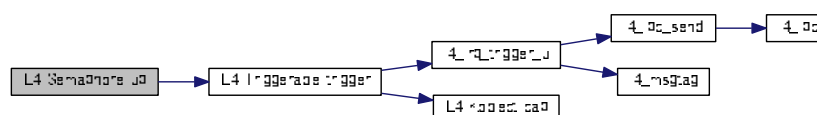
Send-only IPC message return tag. Use [l4_ipc_error\(\)](#) to check for errors, do **not** use [l4_error\(\)](#).

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least $2^{31}-1$.

Definition at line 65 of file [semaphore](#).

References [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



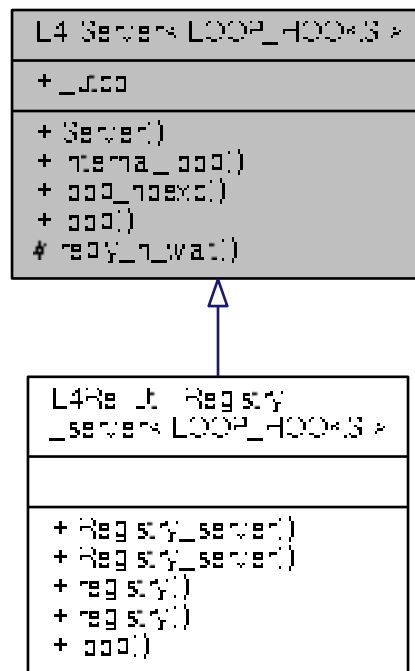
The documentation for this struct was generated from the following file:

- [l4/sys/semaphore](#)

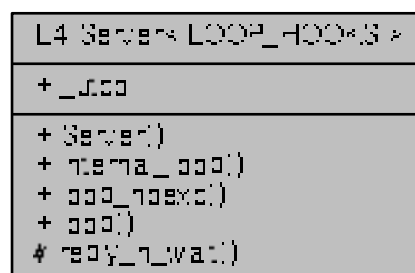
14.164 L4::Server< LOOP_HOOKS > Class Template Reference

Basic server loop for handling client requests.

Inheritance diagram for L4::Server< LOOP_HOOKS >:



Collaboration diagram for L4::Server< LOOP_HOOKS >:



Public Member Functions

- [Server](#) ([l4_utcb_t](#) *utcb)
Initializes the server loop.
- `template<typename DISPATCH >`
`L4_NORETURN void internal_loop (DISPATCH dispatch)`
The server loop.
- `template<typename R >`
`L4_NORETURN void loop_noexc (R r)`
Server loop without exception handling.
- `template<typename EXC , typename R >`
`L4_NORETURN void loop (R r)`
Server loop with internal exception handling.

Protected Member Functions

- `l4_msgtag_t reply_n_wait (l4_msgtag_t reply, l4_umword_t *p)`
Internal implementation for reply and wait.

14.164.1 Detailed Description

```
template<typename LOOP_HOOKS = ipc_svr::Default_loop_hooks>
class L4::Server< LOOP_HOOKS >
```

Basic server loop for handling client requests.

Parameters

<code>LOOP_HOOKS</code>	the server inherits from LOOP_HOOKS and calls the hooks defined in LOOP_HOOKS in the server loop. See ipc_svr::Default_loop_hooks , ipc_svr::Ignore_errors , ipc_svr::Default_timeout , ipc_svr::Compound_reply , and ipc_svr::Br_manager_no_buffers .
-------------------------	--

This is basically a simple server loop that uses a single message buffer for receiving requests and sending replies. The dispatcher determines how incoming messages are handled.

Definition at line 290 of file [ipc_server_loop](#).

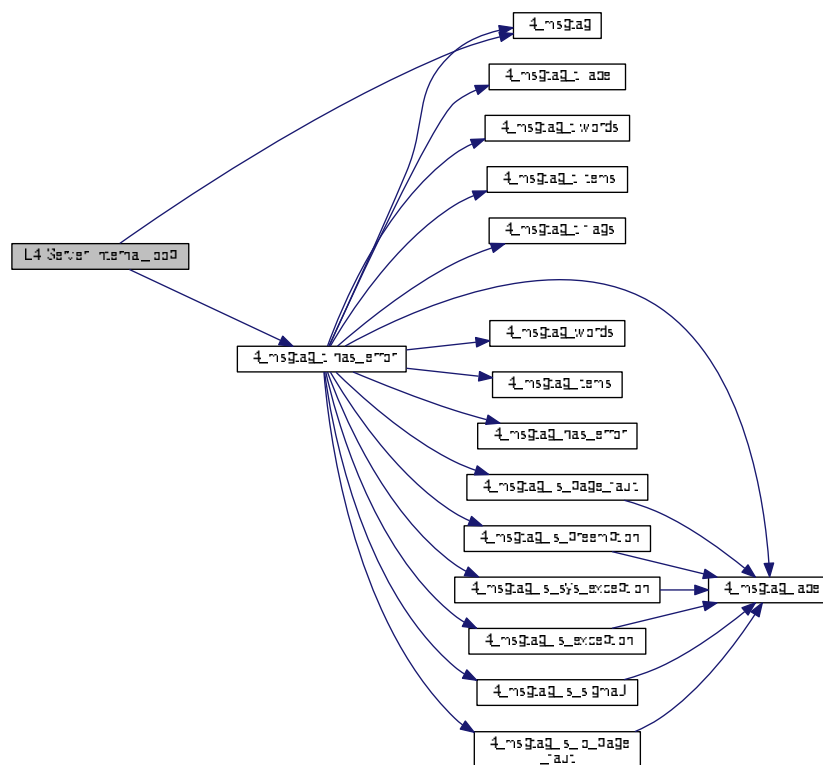
14.164.2 Constructor & Destructor Documentation

14.164.2.1 [Server\(\)](#)

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
    l4\_utcb\_t * utcb ) [inline], [explicit]
```

Initializes the server loop.

<i>utcb</i>	The UTCB of the thread running the server loop.
-------------	---

References [L4_NORETURN.](#)

14.164.3.2 loop()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
template<typename EXC , typename R >
L4_NORETURN void L4::Server< LOOP_HOOKS >::loop (
    R r ) [inline]
```

[Server](#) loop with internal exception handling.

This server loop translates [L4::Runtime_error](#) exceptions into negative error return codes sent to the caller.

Definition at line 323 of file [ipc_server_loop](#).

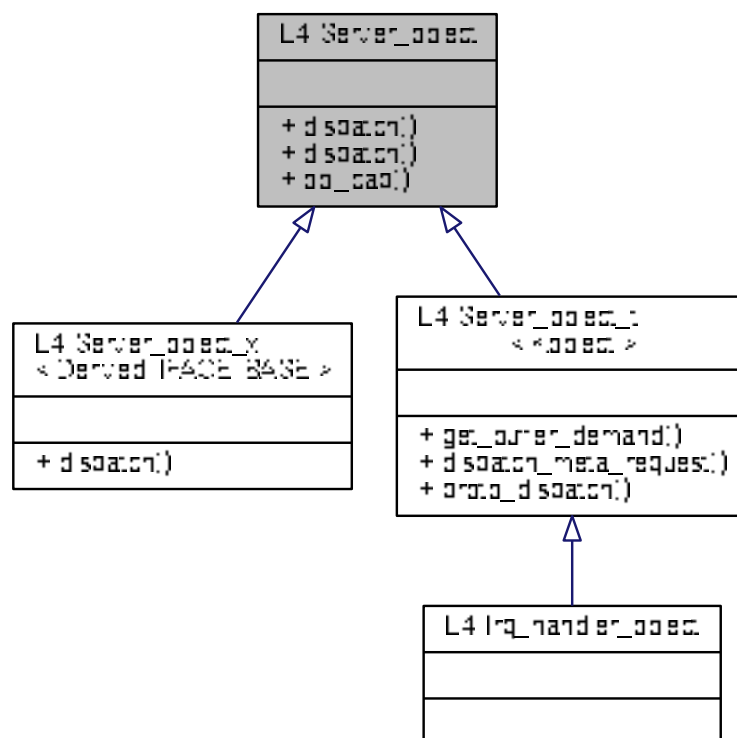
The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_server_loop

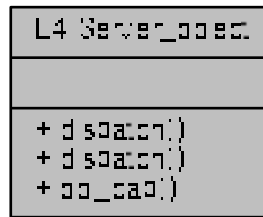
14.165 L4::Server_object Class Reference

Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).

Inheritance diagram for L4::Server_object:



Collaboration diagram for L4::Server_object:



Public Member Functions

- virtual int [dispatch](#) (unsigned long rights, [ipc::lostream](#) &ios)=0
The abstract handler for client requests to the object.

14.165.1 Detailed Description

Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).

Note

Usually [L4::Server_object_t](#) is used as a base class when writing server objects.

This server object provides an abstract interface that is used by the [L4::Registry_dispatcher](#) model. You can derive subclasses from this interface and implement application specific server objects.

Definition at line [49](#) of file [ipc_server](#).

14.165.2 Member Function Documentation

14.165.2.1 dispatch()

```
virtual int L4::Server_object::dispatch (
    unsigned long rights,
    Ipc::lostream & ios ) [pure virtual]
```

The abstract handler for client requests to the object.

Parameters

<i>rights</i>	The rights bits in the invoked capability.
<i>ios</i>	The lpc::lostream for reading the request and writing the reply.

Return values

<code>-L4_ENOREPLY</code>	Instructs the server loop to not send a reply.
<code>< 0</code>	Error, reply with error code.
<code>>= 0</code>	Success, reply with return value.

This function must be implemented by application specific server objects. The implementation must unmarshall data from the stream (`ios`) and create a reply by marshalling to the stream (`ios`). For details about the IPC stream see [IPC stream operators](#).

Note

You need to extract the complete message from the `ios` stream before inserting any reply data or before doing any function call that may use the UTCB. Otherwise, the incoming message may get lost.

Implemented in [L4::Server_object_x< Derived, IFACE, BASE >](#).

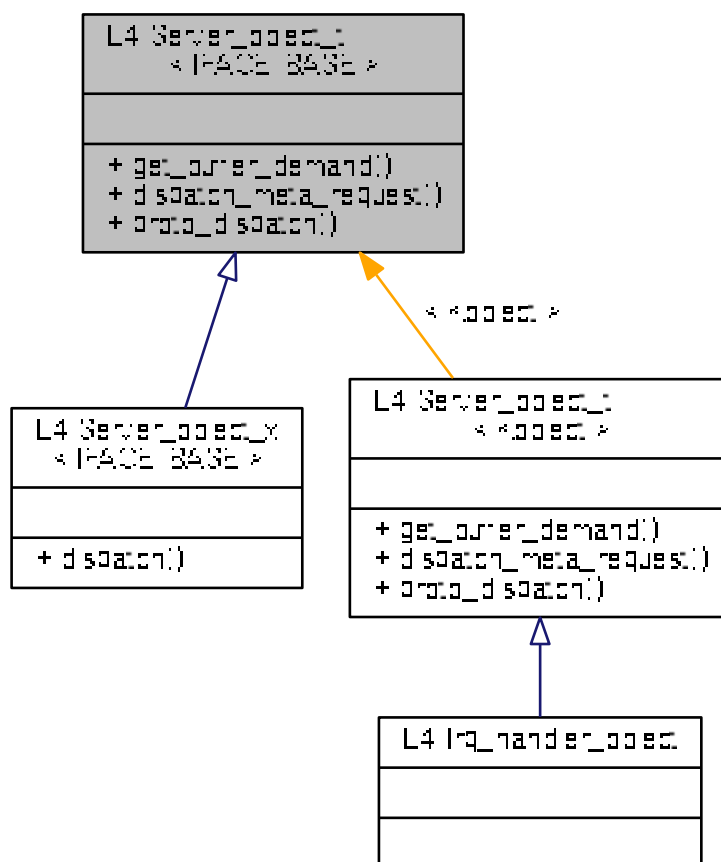
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_server](#)

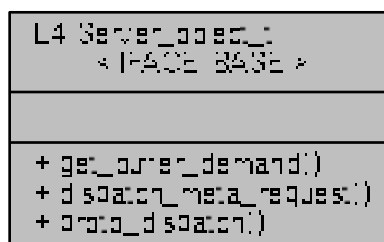
14.166 L4::Server_object_t< IFACE, BASE > Struct Template Reference

Base class (template) for server implementing server objects.

Inheritance diagram for L4::Server_object_t< IFACE, BASE >:



Collaboration diagram for L4::Server_object_t< IFACE, BASE >:



Public Types

- typedef IFACE [Interface](#)
Data type of the IPC interface definition.

Public Member Functions

- BASE::Demand [get_buffer_demand](#) () const
- int [dispatch_meta_request](#) (L4::ipc::Iostream &ios)
Implementation of the meta protocol based on IFACE.

Static Public Member Functions

- template<typename THIS >
static int [proto_dispatch](#) (THIS *self, l4_umword_t rights, L4::ipc::Iostream &ios)
Implementation of protocol-based dispatch for this server object.

14.166.1 Detailed Description

```
template<typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_t< IFACE, BASE >
```

Base class (template) for server implementing server objects.

Template Parameters

<i>IFACE</i>	The IPC interface class that defines the interface that shall be implemented.
<i>BASE</i>	The server object base class (usually L4::Server_object).

Examples:

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 91 of file [ipc_server](#).

14.166.2 Member Function Documentation

14.166.2.1 dispatch_meta_request()

```
template<typename IFACE, typename BASE = L4::Server_object>
int L4::Server_object_t< IFACE, BASE >::dispatch_meta_request (
    L4::Ipc::Iostream & ios ) [inline]
```

Implementation of the meta protocol based on *IFACE*.

Parameters

<i>ios</i>	The IO stream used for receiving the message.
------------	---

This function can be used to handle incoming [L4_PROTO_META](#) protocol requests. The implementation uses the [L4::Type_info](#) of *IFACE* to handle the requests. Call this function in the implementation of [Server_object::dispatch\(\)](#) when the received message tag has protocol [L4_PROTO_META](#) ([L4::Meta::Protocol](#)).

Definition at line 110 of file [ipc_server](#).

14.166.2.2 get_buffer_demand()

```
template<typename IFACE, typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand ( ) const [inline]
```

Returns

the server-side buffer demand based in *IFACE*.

Definition at line 97 of file [ipc_server](#).

14.166.2.3 proto_dispatch()

```
template<typename IFACE, typename BASE = L4::Server_object>
template<typename THIS >
static int L4::Server_object_t< IFACE, BASE >::proto_dispatch (
    THIS * self,
    l4_umword_t rights,
    L4::Ipc::Iostream & ios ) [inline], [static]
```

Implementation of protocol-based dispatch for this server object.

Parameters

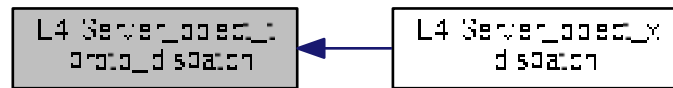
<i>self</i>	The this pointer for the object (inherits from Server_object_t).
<i>rights</i>	The rights from the received IPC (forwarded to p_dispatch()).
<i>ios</i>	The message stream for the incoming and the reply message.

[Server](#) objects may call this function from their [dispatch\(\)](#) function. This function reads the protocol ID from the message tag and uses the [p_dispatch](#) code to dispatch to overloaded [p_dispatch](#) functions of self.

Definition at line 125 of file [ipc_server](#).

Referenced by [L4::Server_object_x< Derived, IFACE, BASE >::dispatch\(\)](#).

Here is the caller graph for this function:



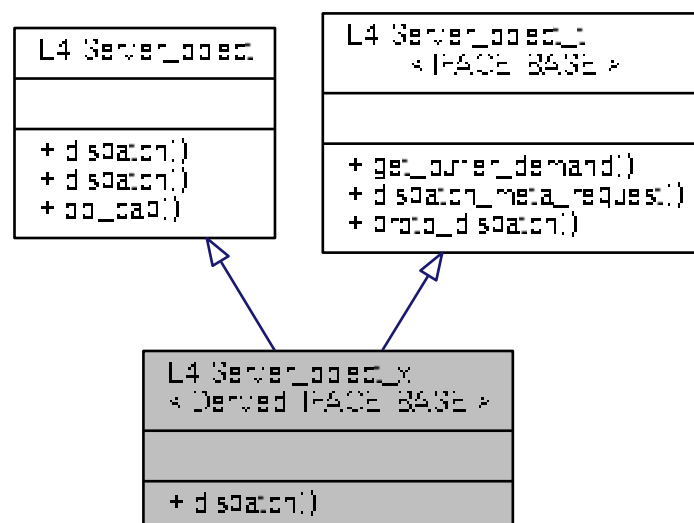
The documentation for this struct was generated from the following file:

- [l4/cxx/ipc_server](#)

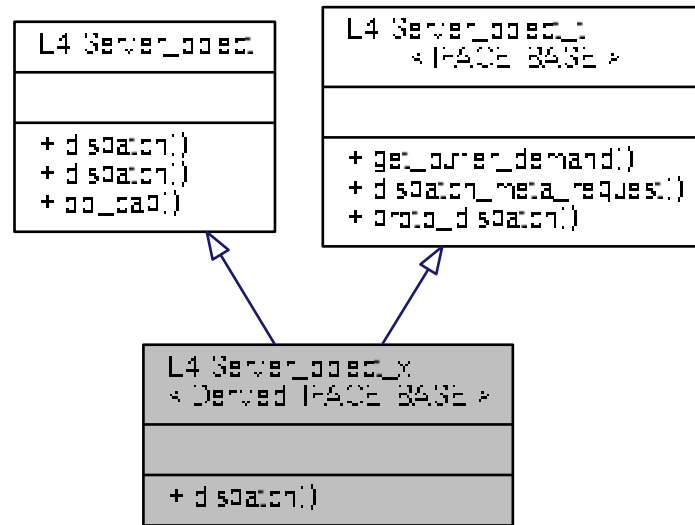
14.167 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference

Helper class to implement p_dispatch based server objects.

Inheritance diagram for L4::Server_object_x< Derived, IFACE, BASE >:



Collaboration diagram for L4::Server_object_x< Derived, IFACE, BASE >:



Public Member Functions

- `int dispatch (l4_umword_t r, L4::lpc::Iostream &ios)`
Implementation forwarding to `p_dispatch()`.

Additional Inherited Members

14.167.1 Detailed Description

```
template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_x< Derived, IFACE, BASE >
```

Helper class to implement `p_dispatch` based server objects.

Template Parameters

<i>Derived</i>	The data type of your server object class.
<i>IFACE</i>	The data type providing the interface definition for the object.
<i>BASE</i>	Optional data-type of the base server object (usually L4::Server_object)

This class implements the standard `dispatch()` function of [L4::Server_object](#) and forwards incoming messages to a set of overloaded `p_dispatch()` functions. There must be a `p_dispatch()` function in *Derived* for each interface provided by *IFACE* with the signature

```
int p_dispatch(Iface *, unsigned rights, L4::lpc::Iostream &)
```

that is called for messages with protocol == `iface::Protocol`.

Example signature for `L4Re::Dataspace` is:

```
int p_dispatch(L4Re::Dataspace *, unsigned, L4::Ipc::Iostream &)
```

Definition at line 154 of file `ipc_server`.

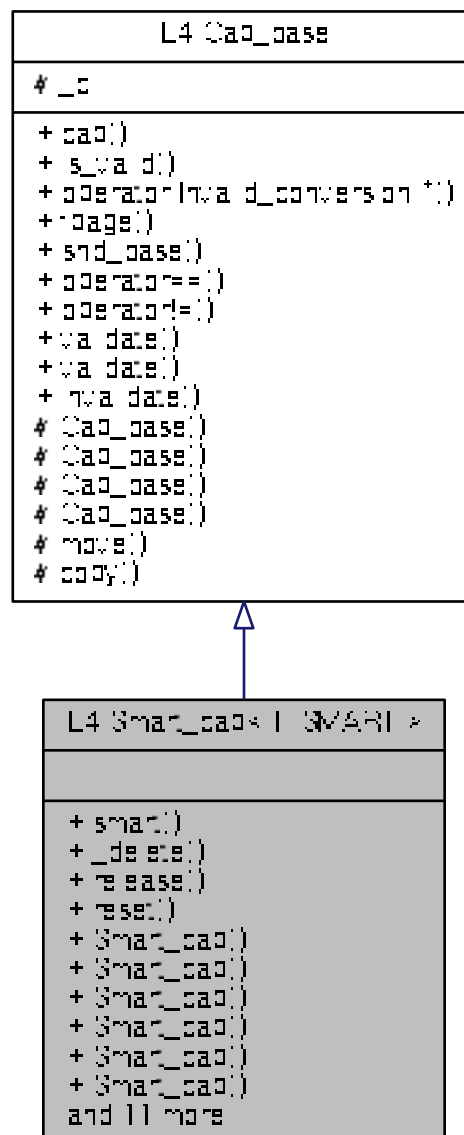
The documentation for this struct was generated from the following file:

- `I4/cxx/ipc_server`

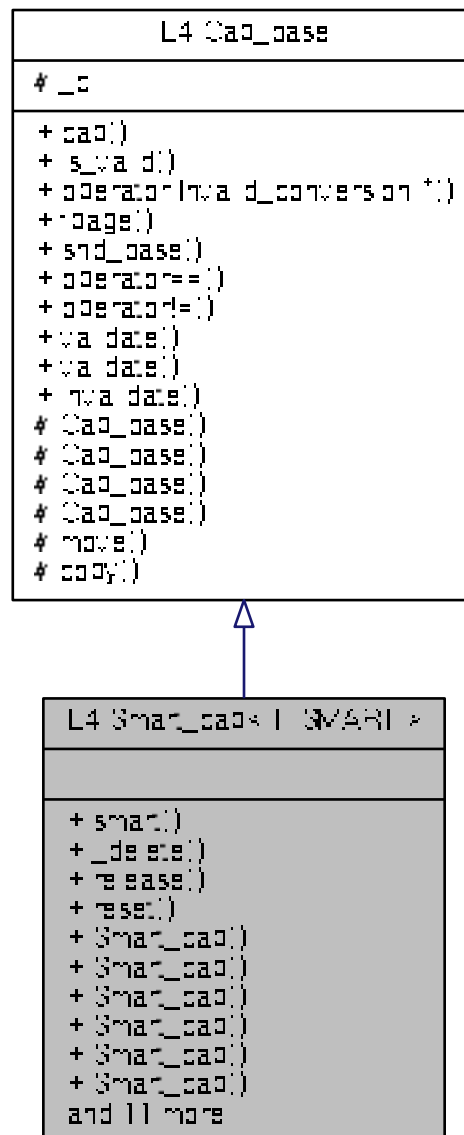
14.168 L4::Smart_cap< T, SMART > Class Template Reference

Smart capability class.

Inheritance diagram for L4::Smart_cap< T, SMART >:



Collaboration diagram for L4::Smart_cap< T, SMART >:



Public Member Functions

- `template<typename O >`
`Smart_cap (Cap< O > const &p) throw ()`
Internal constructor, use to generate a capability from a `this` pointer.
- `Cap< T > operator-> () const throw ()`
Member access of a `T`.

Additional Inherited Members

14.168.1 Detailed Description

```
template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >
```

Smart capability class.

Definition at line 36 of file [smart_capability](#).

14.168.2 Constructor & Destructor Documentation

14.168.2.1 Smart_cap()

```
template<typename T, typename SMART>
template<typename O >
L4::Smart_cap< T, SMART >::Smart_cap (
    Cap< O > const & p ) throw ()    [inline]
```

Internal constructor, use to generate a capability from a `this` pointer.

Attention

This constructor is only useful to generate a capability from the `this` pointer of an objected that is an [L4::Kobject](#). Do never use this constructor for something else!

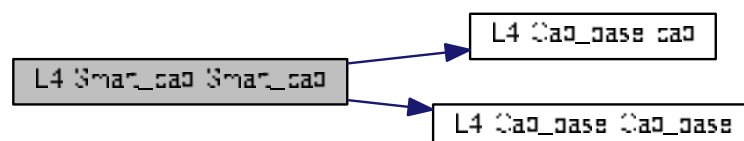
Parameters

<i>p</i>	The <code>this</code> pointer of the Kobject or derived object
----------	--

Definition at line 73 of file [smart_capability](#).

References [L4::Cap_base::_c](#), [L4::Cap_base::cap\(\)](#), and [L4::Cap_base::Cap_base\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

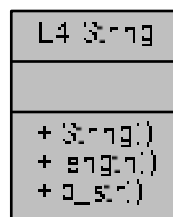
- [l4/sys/smart_capability](#)

14.169 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:



14.169.1 Detailed Description

A null-terminated string container class.

Definition at line 33 of file [string.h](#).

The documentation for this class was generated from the following file:

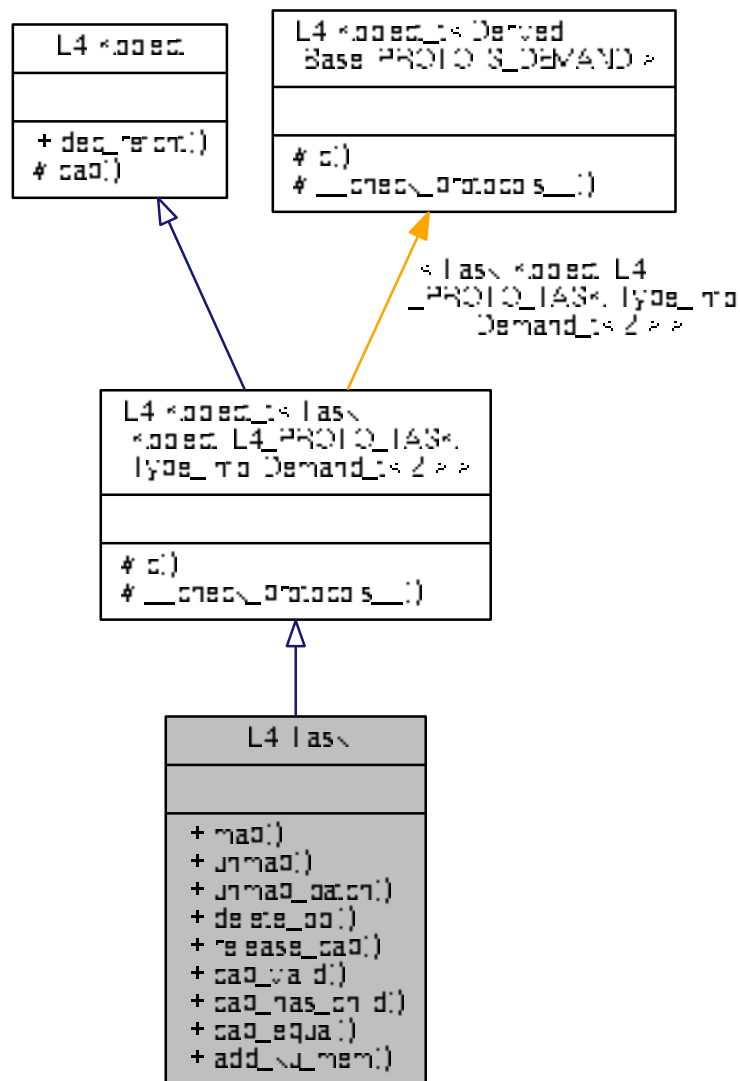
- [l4/cxx/string.h](#)

14.170 L4::Task Class Reference

C++ interface of the [Task](#) kernel object.



Collaboration diagram for L4::Task:



Public Member Functions

- `l4_msgtag_t map (Cap< Task > const &src_task, l4_fpage_t const &snd_fpage, l4_addr_t snd_base, l4_utcb_t *utcb=l4_utcb()) throw ()`
Map resources available in the source task to a destination task.
- `l4_msgtag_t unmap (l4_fpage_t const &fpage, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) throw ()`
Revoke rights from the task.
- `l4_msgtag_t unmap_batch (l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) throw ()`
Revoke rights from a task.
- `l4_msgtag_t delete_obj (L4::Cap< void > obj, l4_utcb_t *utcb=l4_utcb()) throw ()`
Release capability and delete object.

- `l4_msgtag_t release_cap (L4::Cap< void > cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
Release capability.
- `l4_msgtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
Check whether a capability is present (refers to an object).
- `l4_msgtag_t cap_has_child (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
Test whether a capability has child mappings (in another task).
- `l4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) throw ()`
Test whether two capabilities point to the same object with the same rights.
- `l4_msgtag_t add_ku_mem (l4_fpage_t const &fpage, l4_utcb_t *utcb=l4_utcb()) throw ()`
Add kernel-user memory.

Additional Inherited Members

14.170.1 Detailed Description

C++ interface of the [Task](#) kernel object.

The [L4::Task](#) class represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space associated with an [L4::Task](#).

[L4::Task](#) objects are created using the [L4::Factory](#) interface.

Include File

```
#include <l4/sys/task>
```

Definition at line 43 of file [task](#).

14.170.2 Member Function Documentation

14.170.2.1 add_ku_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
    l4_fpage_t const & fpage,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Add kernel-user memory.

Parameters

<i>fpage</i>	Flexpage describing the virtual area the memory goes to.
<i>utcb</i>	UTCP pointer of the calling thread.

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page (`L4_PAGESIZE`). A portable implementation should not depend on allocations greater than 16KiB to succeed.

Returns

Syscall return tag

Definition at line 219 of file [task](#).

14.170.2.2 cap_equal()

```
l4_msgtag_t L4::Task::cap_equal (
    Cap< void > const & cap_a,
    Cap< void > const & cap_b,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Test whether two capabilities point to the same object with the same rights.

Parameters

<i>cap_a</i>	First capability selector to compare.
<i>cap_b</i>	Second capability selector to compare.
<i>utcb</i>	Optional: UTCB pointer of the calling thread.

Return values

<i>tag.label()</i>	= 1: <i>cap_a</i> and <i>cap_b</i> point to the same object.
<i>tag.label()</i>	= 0: The two caps do not point to the same object.

Definition at line 200 of file [task](#).

14.170.2.3 cap_has_child()

```
l4_msgtag_t L4::Task::cap_has_child (
    Cap< void > const & cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Test whether a capability has child mappings (in another task).

Parameters

<i>cap</i>	Capability selector to look up in the destination task.
<i>utcb</i>	UTCB pointer of the calling thread.

Return values

<i>tag.label()</i>	= 1: The capability has at least on child mapping in another task.
<i>tag.label()</i>	= 0: No child mappings.

Deprecated Do not use. Undetermined future, might be removed.

Definition at line 183 of file [task](#).

14.170.2.4 cap_valid()

```
l4_msgtag_t L4::Task::cap_valid (
    Cap< void > const & cap,
    l4_utcb_t * utcb = l4_utcb() ) throw () [inline]
```

Check whether a capability is present (refers to an object).

Parameters

<i>cap</i>	Valid capability to check for presence.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Precondition

`cap` must be a valid capability (i.e. `cap.is_valid() == true`). If you are unsure about the validity of your capability use [L4::Cap.validate\(\)](#) instead.

Definition at line 167 of file [task](#).

14.170.2.5 delete_obj()

```
l4_msgtag_t L4::Task::delete_obj (
    L4::Cap< void > obj,
    l4_utcb_t * utcb = l4_utcb() ) throw () [inline]
```

Release capability and delete object.

Parameters

<i>obj</i>	Capability selector of the object to delete.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag

The object will be deleted if the `obj` has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

Definition at line 133 of file [task](#).

14.170.2.6 `map()`

```
l4_msgtag_t L4::Task::map (
    Cap< Task > const & src_task,
    l4_fpage_t const & snd_fpage,
    l4_addr_t snd_base,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Map resources available in the source task to a destination task.

Parameters

<i>src_task</i>	Capability selector of the source task.
<i>snd_fpage</i>	Send flexpage that describes an area in the address space or object space of the source task.
<i>snd_base</i>	Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects. The destination task is the task referenced by the capability invoking `map` and the receive window is the whole address space of said task.

Definition at line 67 of file [task](#).

14.170.2.7 `release_cap()`

```
l4_msgtag_t L4::Task::release_cap (
    L4::Cap< void > cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Release capability.

Parameters

<i>cap</i>	Capability selector to release.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This operation unmaps the capability from `this` task.

Definition at line 147 of file [task](#).

14.170.2.8 unmap()

```
l4_msgtag_t L4::Task::unmap (
    l4_fpage_t const & fpage,
    l4_umword_t map_mask,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Revoke rights from the task.

Parameters

<i>fpage</i>	Flexpage that describes an area in the address space or object space of <code>this</code> task
<i>map_mask</i>	Unmap mask, see l4_unmap_flags_t
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line 90 of file [task](#).

Referenced by [L4Re::Rm::attach\(\)](#).

Here is the caller graph for this function:



14.170.2.9 unmap_batch()

```

l4_msgtag_t L4::Task::unmap_batch (
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_umword_t map_mask,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

Revoke rights from a task.

Parameters

<i>fpages</i>	An array of flexpages. Each item describes an area in the address or object space of <i>this</i> task.
<i>num_fpages</i>	Number of fpages in the <i>fpages</i> array.
<i>map_mask</i>	Unmap mask, see l4_unmap_flags_t .
<i>utcb</i>	UTCB pointer of the calling thread.

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

The caller needs to take care that *num_fpages* is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line [115](#) of file [task](#).

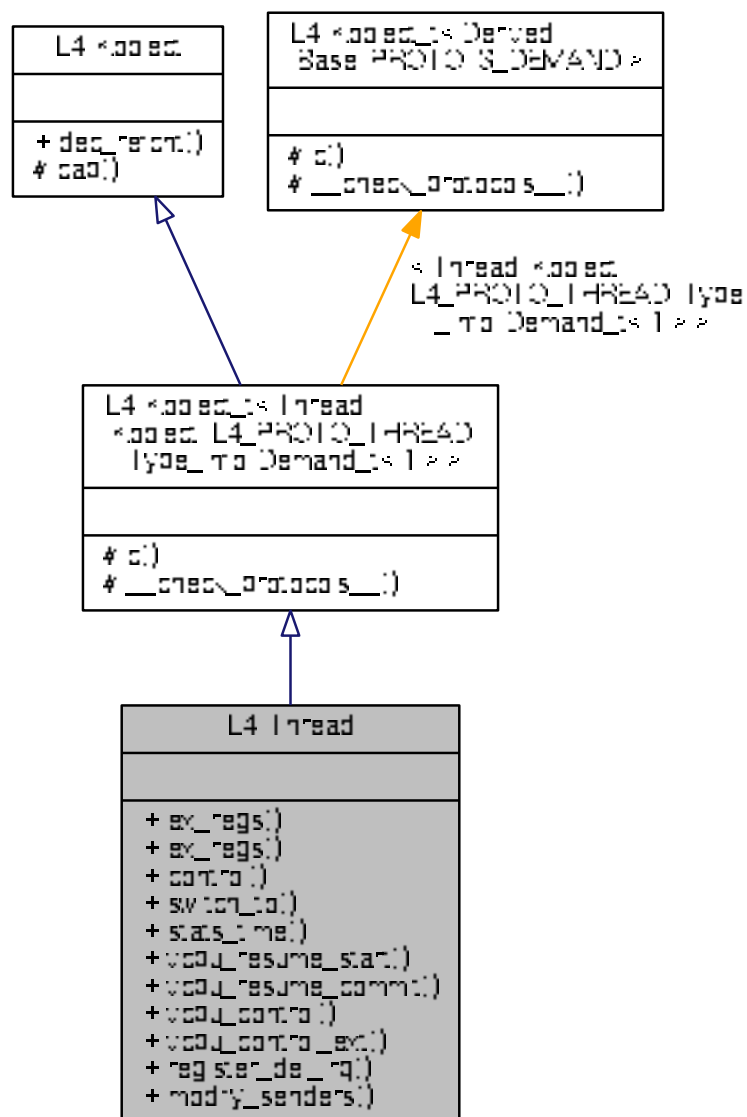
The documentation for this class was generated from the following file:

- [l4/sys/task](#)

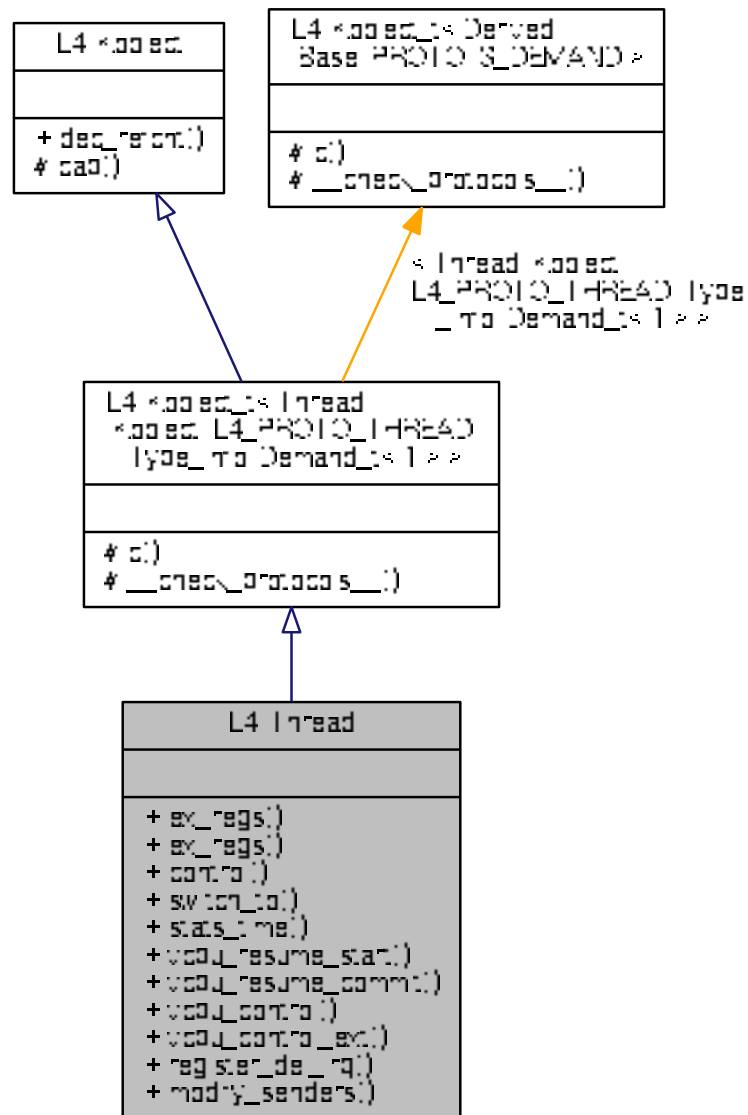
14.171 L4::Thread Class Reference

C++ L4 kernel thread interface.

Inheritance diagram for L4::Thread:



Collaboration diagram for L4::Thread:



Data Structures

- class [Attr](#)
Thread attributes used for control_commit().
- class [Modify_senders](#)
Wrapper class for modifying senders.

Public Member Functions

- [l4_msgtag_t ex_regs](#) ([l4_addr_t](#) ip, [l4_addr_t](#) sp, [l4_umword_t](#) flags, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()

- *Exchange basic thread registers.*
[l4_msgtag_t ex_regs](#) ([l4_addr_t](#) *ip, [l4_addr_t](#) *sp, [l4_umword_t](#) *flags, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Exchange basic thread registers and return previous values.
- [l4_msgtag_t control](#) ([Attr](#) const &attr) throw ()
Commit the given thread-attributes object.
- [l4_msgtag_t switch_to](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Switch execution to this thread.
- [l4_msgtag_t stats_time](#) ([l4_kernel_clock_t](#) *us, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Get consumed time of thread in us.
- [l4_msgtag_t vcpu_resume_start](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
vCPU resume, start.
- [l4_msgtag_t vcpu_resume_commit](#) ([l4_msgtag_t](#) tag, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
vCPU resume, commit.
- [l4_msgtag_t vcpu_control](#) ([l4_addr_t](#) vcpu_state, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Enable or disable the vCPU feature for the thread.
- [l4_msgtag_t vcpu_control_ext](#) ([l4_addr_t](#) ext_vcpu_state, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Enable or disable the extended vCPU feature for the thread.
- [l4_msgtag_t register_del_irq](#) ([Cap](#) < [Irq](#) > [irq](#), [l4_utcb_t](#) *u=[l4_utcb\(\)](#)) throw ()
Register an IRQ that will trigger upon deletion events.
- [l4_msgtag_t modify_senders](#) ([Modify_senders](#) const &todo) throw ()
Apply sender modification rules.

Additional Inherited Members

14.171.1 Detailed Description

C++ [L4](#) kernel thread interface.

The [Thread](#) class defines a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. [Thread](#) kernel objects are created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

Amongst other things an [L4::Thread](#) encapsulates:

- CPU state
 - General-purpose registers
 - Program counter
 - Stack pointer
- FPU state
- Scheduling parameters, see the [L4::Scheduler](#) API
- Execution state
 - Blocked, Runnable, Running

[Thread](#) objects provide an API for

- [Thread](#) configuration and manipulation
- [Thread](#) switching.

Include File

```
#include <l4/sys/thread>
```

For the C interface see the [Thread](#) API.

Definition at line 58 of file [thread](#).

14.171.2 Member Function Documentation

14.171.2.1 control()

```
l4_msgtag_t L4::Thread::control (
    Attr const & attr ) throw ()    [inline]
```

Commit the given thread-attributes object.

Parameters

<i>attr</i>	the attribute object to commit to the thread.
-------------	---

Definition at line 215 of file [thread](#).

14.171.2.2 ex_regs() [1/2]

```
l4_msgtag_t L4::Thread::ex_regs (
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Exchange basic thread registers.

Parameters

<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags .
<i>utcb</i>	UTCB to use for this operation.

Returns

System call return tag

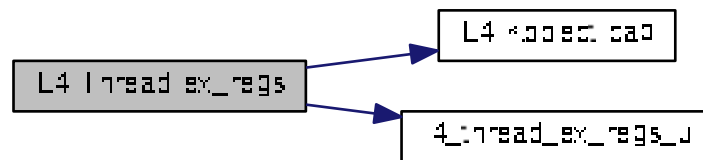
This method allows to manipulate a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

The thread is started using [L4::Scheduler::run_thread\(\)](#). However, if at the time [L4::Scheduler::run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [ex_regs\(\)](#) with a valid instruction pointer might start the thread.

Definition at line 85 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_ex_regs_u\(\)](#).

Here is the call graph for this function:



14.171.2.3 ex_regs() [2/2]

```

l4_msgtag_t L4::Thread::ex_regs (
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Exchange basic thread registers and return previous values.

Parameters

in, out	<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see L4_thread_ex_regs_flags , return previous CPU flags of the thread.
	<i>utcb</i>	UTCB to use for this operation.

Returns

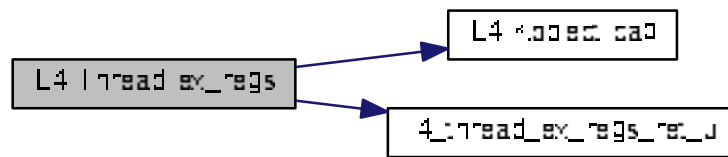
System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

Definition at line 111 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_ex_regs_ret_u\(\)](#).

Here is the call graph for this function:



14.171.2.4 modify_senders()

```

l4_msgtag_t L4::Thread::modify_senders (
    Modify_senders const & todo ) throw ()    [inline]
  
```

Apply sender modification rules.

Parameters

<i>todo</i>	Prepared sender modification rules.
-------------	-------------------------------------

Returns

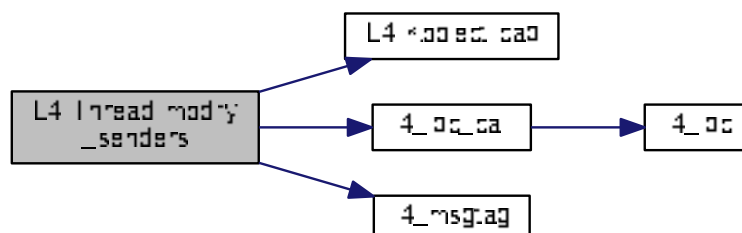
System call return tag.

Definition at line 373 of file [thread](#).

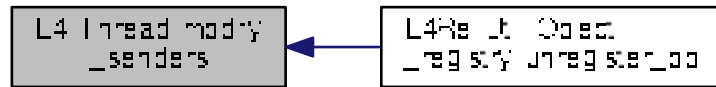
References [L4::Kobject::cap\(\)](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), and [L4_PROTO_THREAD](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.171.2.5 register_del_irq()

```
l4_msgtag_t L4::Thread::register_del_irq (
    Cap< Irq > irq,
    l4_utcb_t * u = l4_utcb() ) throw ()    [inline]
```

Register an IRQ that will trigger upon deletion events.

Parameters

<i>irq</i>	Capability selector for the IRQ object to be triggered.
<i>u</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

System call return tag containing the return code.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

See also

[l4_thread_register_del_irq](#)

Definition at line 313 of file [thread](#).

14.171.2.6 stats_time()

```
l4_msgtag_t L4::Thread::stats_time (
    l4_kernel_clock_t * us,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get consumed time of thread in us.

Parameters

out	us	Consumed time in μ s.
	utcb	UTCB of the current thread.

Returns

Syscall return tag.

Definition at line 236 of file [thread](#).

14.171.2.7 switch_to()

```
l4_msgtag_t L4::Thread::switch_to (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Switch execution to this thread.

Parameters

utcb	the UTCB of the current thread.
------	---------------------------------

Note

The current time slice is inherited to this thread.

Definition at line 225 of file [thread](#).

14.171.2.8 vcpu_control()

```
l4_msgtag_t L4::Thread::vcpu_control (
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Enable or disable the vCPU feature for the thread.

Parameters

vcpu_state	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
utcb	UTCB to use for this operation.

Returns

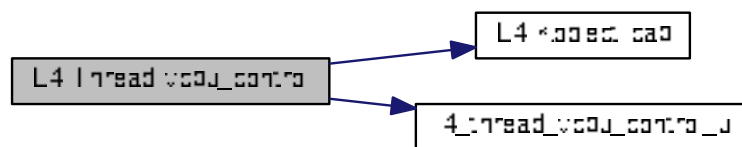
Syscall return tag.

This function enables the vCPU feature of `this` thread if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 272 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_u\(\)](#).

Here is the call graph for this function:

**14.171.2.9 vcpu_control_ext()**

```
l4_msgtag_t L4::Thread::vcpu_control_ext (
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

Enable or disable the extended vCPU feature for the thread.

Parameters

<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()).
<i>utcb</i>	UTCB to use for this operation.

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of `this` thread if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

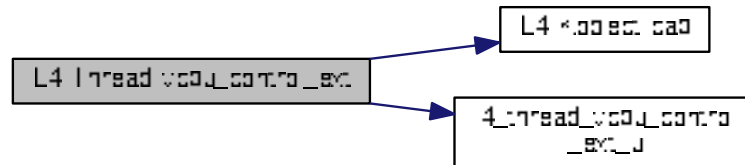
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 296 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_ext_u\(\)](#).

Here is the call graph for this function:

14.171.2.10 `vcpu_resume_commit()`

```
l4_msgtag_t L4::Thread::vcpu_resume_commit (
    l4_msgtag_t tag,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

vCPU resume, commit.

See also

[l4_thread_vcpu_resume_commit](#)

Definition at line 253 of file [thread](#).

14.171.2.11 `vcpu_resume_start()`

```
l4_msgtag_t L4::Thread::vcpu_resume_start (
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

vCPU resume, start.

See also

[l4_thread_vcpu_resume_start](#)

Definition at line 245 of file [thread](#).

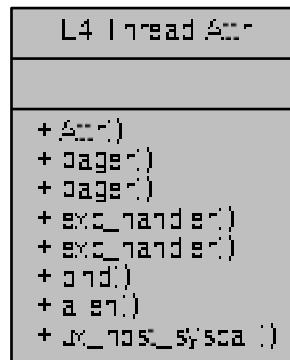
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

14.172 L4::Thread::Attr Class Reference

[Thread](#) attributes used for `control_commit()`.

Collaboration diagram for L4::Thread::Attr:



Public Member Functions

- [Attr](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Create a thread-attribute object with the given UTCB.
- void [pager](#) ([Cap](#)< void > const &pager) throw ()
Set the pager capability selector.
- [Cap](#)< void > [pager](#) () throw ()
Get the capability selector used for page-fault messages.
- void [exc_handler](#) ([Cap](#)< void > const &exc_handler) throw ()
Set the exception-handler capability selector.
- [Cap](#)< void > [exc_handler](#) () throw ()
Get the capability selector used for exception messages.
- void [bind](#) ([l4_utcb_t](#) *thread_utcb, [Cap](#)< [Task](#) > const &task) throw ()
Bind the thread to a task.
- void [alien](#) (int on) throw ()
Set the thread to alien mode.
- void [ux_host_syscall](#) (int on) throw ()
Allow host system calls on Fiasco-UX.

Friends

- class [L4::Thread](#)

14.172.1 Detailed Description

[Thread](#) attributes used for `control_commit()`.

This class is responsible for initializing various attributes of a thread in a UTCB for the `control_commit()` method.

See also

[Thread control](#) for some more details.

Definition at line 125 of file [thread](#).

14.172.2 Constructor & Destructor Documentation

14.172.2.1 Attr()

```
L4::Thread::Attr::Attr (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline], [explicit]
```

Create a thread-attribute object with the given UTCB.

Parameters

<i>utcb</i>	The UTCB to use for the later <code>L4::Thread::control_commit()</code> function. Usually this is the UTCB of the calling thread.
-------------	---

Definition at line 138 of file [thread](#).

14.172.3 Member Function Documentation

14.172.3.1 bind()

```
void L4::Thread::Attr::bind (
    l4_utcb_t * thread_utcb,
    Cap< Task > const & task ) throw ()    [inline]
```

Bind the thread to a task.

Parameters

<i>thread_utcb</i>	The UTCB address of the thread within the task specified by <i>task</i> .
<i>task</i>	The capability selector for the task the thread shall be bound to.

Binding a thread to a task means that the thread shall afterwards execute in the given task. To actually start execution you need to use [L4::Thread::ex_regs\(\)](#).

Definition at line [191](#) of file [thread](#).

14.172.3.2 `exc_handler()` [1/2]

```
void L4::Thread::Attr::exc_handler (
    Cap< void > const & exc_handler ) throw ()    [inline]
```

Set the exception-handler capability selector.

Parameters

<i>exc_handler</i>	The capability selector that shall be used for exception messages. This capability selector must be valid within the task the thread is bound to.
--------------------	---

Definition at line [167](#) of file [thread](#).

14.172.3.3 `exc_handler()` [2/2]

```
Cap<void> L4::Thread::Attr::exc_handler ( ) throw ()    [inline]
```

Get the capability selector used for exception messages.

Returns

The capability selector used to send exception messages. The selector is valid in the task the thread is bound to.

Definition at line [176](#) of file [thread](#).

14.172.3.4 `pager()` [1/2]

```
void L4::Thread::Attr::pager (
    Cap< void > const & pager ) throw ()    [inline]
```

Set the pager capability selector.

Parameters

<i>pager</i>	The capability selector that shall be used for page-fault messages. This capability selector must be valid within the task the thread is bound to.
--------------	--

Definition at line 148 of file [thread](#).

14.172.3.5 pager() [2/2]

```
Cap<void> L4::Thread::Attr::pager ( ) throw ( ) [inline]
```

Get the capability selector used for page-fault messages.

Returns

The capability selector used to send page-fault messages. The selector is valid in the task the thread is bound to.

Definition at line 157 of file [thread](#).

14.172.3.6 ux_host_syscall()

```
void L4::Thread::Attr::ux_host_syscall (
    int on ) throw ( ) [inline]
```

Allow host system calls on Fiasco-UX.

Precondition

Running on Fiasco-UX.

Definition at line 205 of file [thread](#).

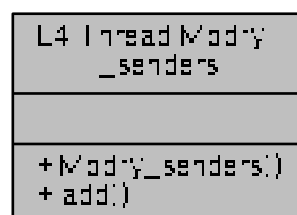
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

14.173 L4::Thread::Modify_senders Class Reference

Wrapper class for modifying senders.

Collaboration diagram for L4::Thread::Modify_senders:



Public Member Functions

- `int add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits) throw ()`

Add a rule.

14.173.1 Detailed Description

Wrapper class for modifying senders.

Use the `add()` function to add modification rules, and use `modify_senders()` to commit. Do not use the UTCB in between as it is used by `add()` and `modify_senders()`.

Definition at line 323 of file `thread`.

14.173.2 Member Function Documentation

14.173.2.1 add()

```
int L4::Thread::Modify_senders::add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits ) throw ()    [inline]
```

Add a rule.

Parameters

<i>match_mask</i>	Bitmask of bits to match the label.
<i>match</i>	Bitmask that must be equal to the label after applying <i>match_mask</i> .
<i>del_bits</i>	Bits to be deleted from the label.
<i>add_bits</i>	Bits to be added to the label.

Returns

0 on success, <0 on error

Only the first match is applied.

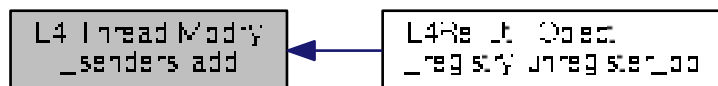
See also

[l4_thread_modify_sender_add\(\)](#)

Definition at line 352 of file `thread`.

Referenced by `L4Re::Util::Object_registry::unregister_obj()`.

Here is the caller graph for this function:



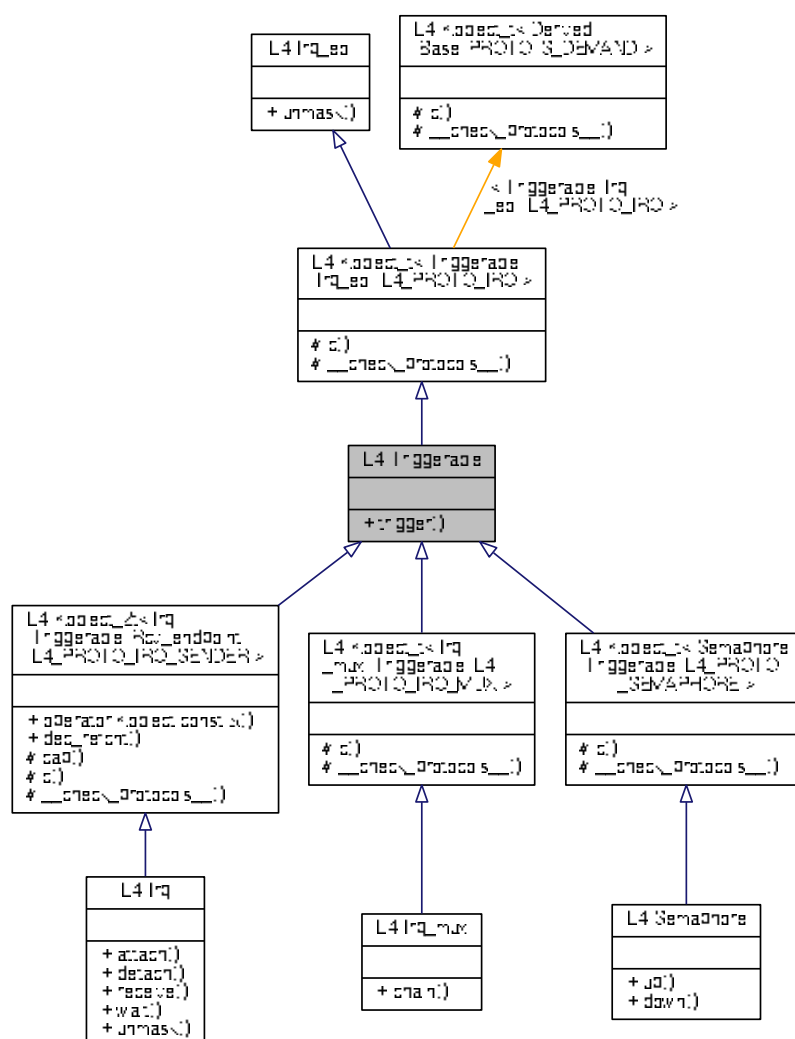
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

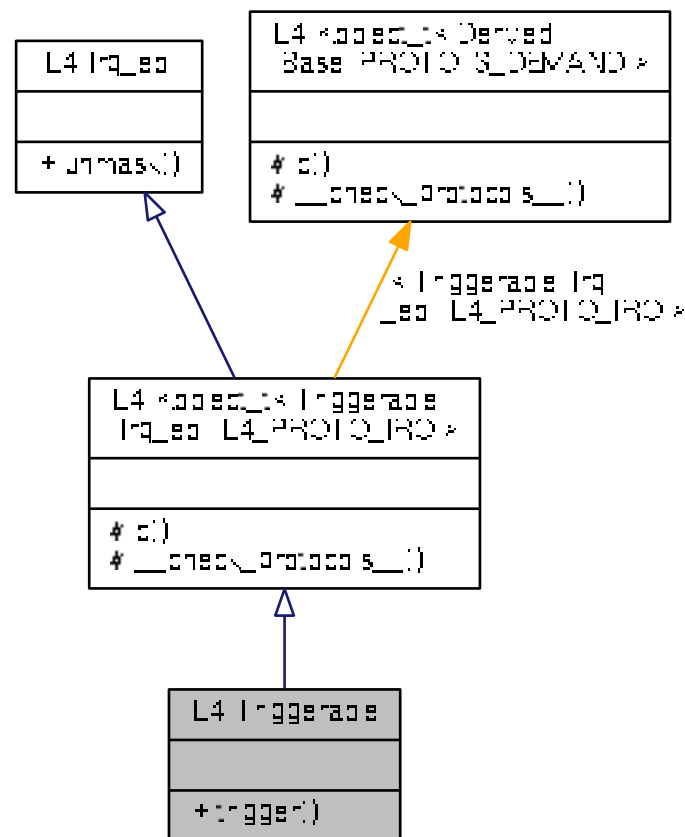
14.174 L4::Triggerable Struct Reference

Interface that allows an object to be triggered by some source.

Inheritance diagram for L4::Triggerable:



Collaboration diagram for L4::Triggerable:



Public Member Functions

- `l4_msgtag_t trigger (l4_utcb_t *utcb=l4_utcb()) throw ()`
Trigger.

Additional Inherited Members

14.174.1 Detailed Description

Interface that allows an object to be triggered by some source.

This interface is usually used in conjunction with [L4::lcu](#).

Definition at line 78 of file [irq](#).

14.174.2 Member Function Documentation

14.174.2.1 trigger()

```
l4_msgtag_t L4::Triggerable::trigger (
    l4_utcb_t * utcb = l4_utcb() ) throw () [inline]
```

Trigger.

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

This function is a send-only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Examples:

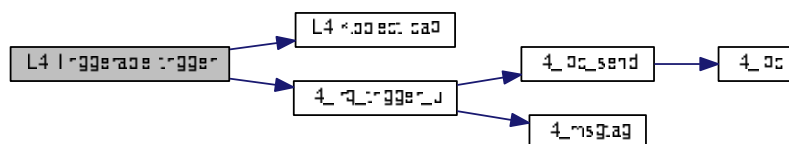
[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 93 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_irq_trigger_u\(\)](#).

Referenced by [L4::Semaphore::up\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

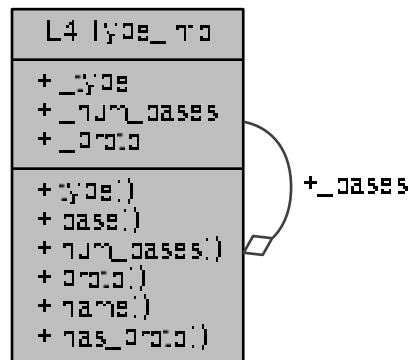
- [l4/sys/irq](#)

14.175 L4::Type_info Struct Reference

Dynamic Type Information for [L4Re](#) Interfaces.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Type_info:



Data Structures

- class [Demand](#)
Data type for expressing the needed receive buffers at the server-side of an interface.
- struct [Demand_t](#)
Template type statically describing demand of receive buffers.
- struct [Demand_union_t](#)
Template type statically describing the combination of two [Demand](#) object.

14.175.1 Detailed Description

Dynamic Type Information for [L4Re](#) Interfaces.

This class represents the runtime-dynamic type information for [L4Re](#) interfaces, and is not intended to be used directly by applications.

Note

The interface of is subject to changes.

The main use for this info is to be used by the implementation of the [L4::cap_dynamic_cast\(\)](#) function.

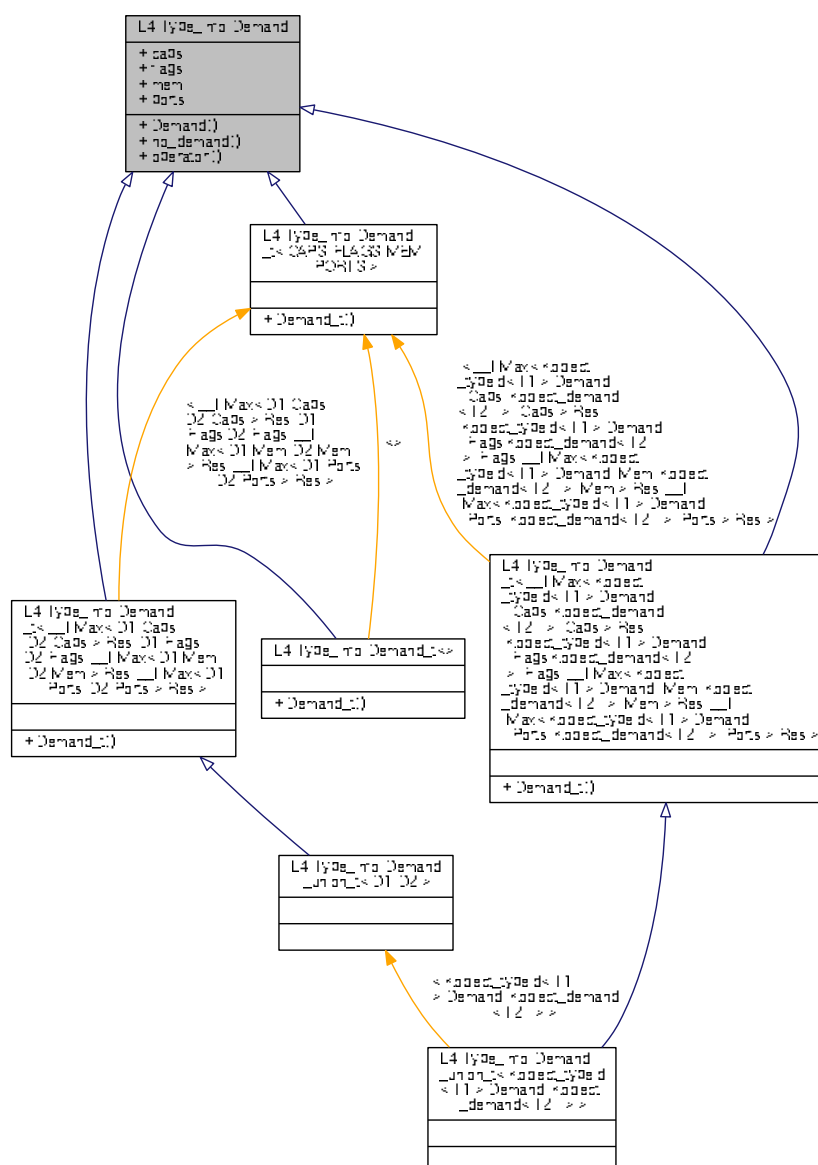
Definition at line 509 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

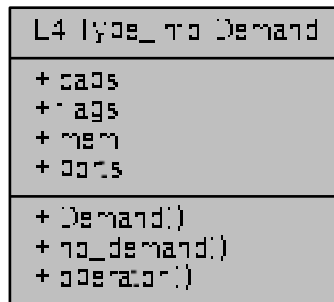
- [l4/sys/__typeinfo.h](#)

Data type for expressing the needed receive buffers at the server-side of an interface.

Inheritance diagram for L4::Type_info::Demand:



Collaboration diagram for L4::Type_info::Demand:



Public Member Functions

- [Demand](#) (unsigned char [caps](#)=0, unsigned char [flags](#)=0, unsigned char [mem](#)=0, unsigned char [ports](#)=0)
Make [Demand](#) object.
- bool [no_demand](#) () const
- [Demand operator|](#) ([Demand](#) const &rhs) const
get the combined demand of this and rhs

Data Fields

- unsigned char [caps](#)
number of capability receive buffers.
- unsigned char [flags](#)
flags, such as the need for timeouts (TBD).
- unsigned char [mem](#)
number of memory receive buffers.
- unsigned char [ports](#)
number of IO-port receive buffers.

14.176.1 Detailed Description

Data type for expressing the needed receive buffers at the server-side of an interface.

Definition at line 516 of file [__typeinfo.h](#).

14.176.2 Constructor & Destructor Documentation

14.176.2.1 Demand()

```
L4::Type_info::Demand::Demand (
    unsigned char caps = 0,
    unsigned char flags = 0,
    unsigned char mem = 0,
    unsigned char ports = 0 ) [inline], [explicit]
```

Make [Demand](#) object.

Parameters

<i>caps</i>	number of capability receive buffers
<i>flags</i>	flags, such as the need for timeouts (TBD).
<i>mem</i>	number of memory receive windows.
<i>ports</i>	number of IO-port receive windows.

Definition at line 537 of file [__typeinfo.h](#).

14.176.3 Member Function Documentation

14.176.3.1 no_demand()

```
bool L4::Type_info::Demand::no_demand ( ) const [inline]
```

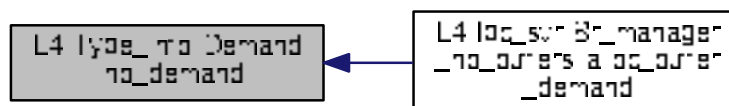
Returns

true if there is no demand at all

Definition at line 542 of file [__typeinfo.h](#).

Referenced by [L4::lpc_svr::Br_manager_no_buffers::alloc_buffer_demand\(\)](#).

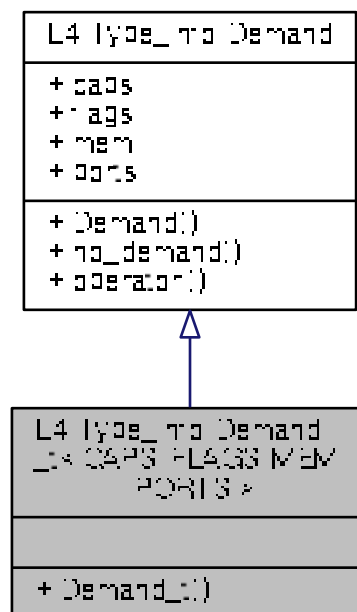
Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/__typeinfo.h](#)

Collaboration diagram for L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >:



Public Types

- enum { `Caps` = CAPS, `Flags` = FLAGS, `Mem` = MEM, `Ports` = PORTS }

Additional Inherited Members

14.177.1 Detailed Description

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char PORTS = 0>
struct L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >
```

Template type statically describing demand of receive buffers.

Template Parameters

<i>CAPS</i>	number of capability receive buffers needed.
<i>FLAGS</i>	flags, such as the need for timeouts (TBD).
<i>MEM</i>	number of memory receive windows needed.
<i>PORTS</i>	number of IO-port receive windwows needed.

Definition at line 563 of file `__typeinfo.h`.

14.177.2 Member Enumeration Documentation

14.177.2.1 anonymous enum

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char  
PORTS = 0>  
anonymous enum
```

Enumerator

Caps	number of capability receive buffers.
Flags	flags, such as the need for timeouts.
Mem	number of memory receive windows.
Ports	number of IO-port receive windows.

Definition at line 565 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

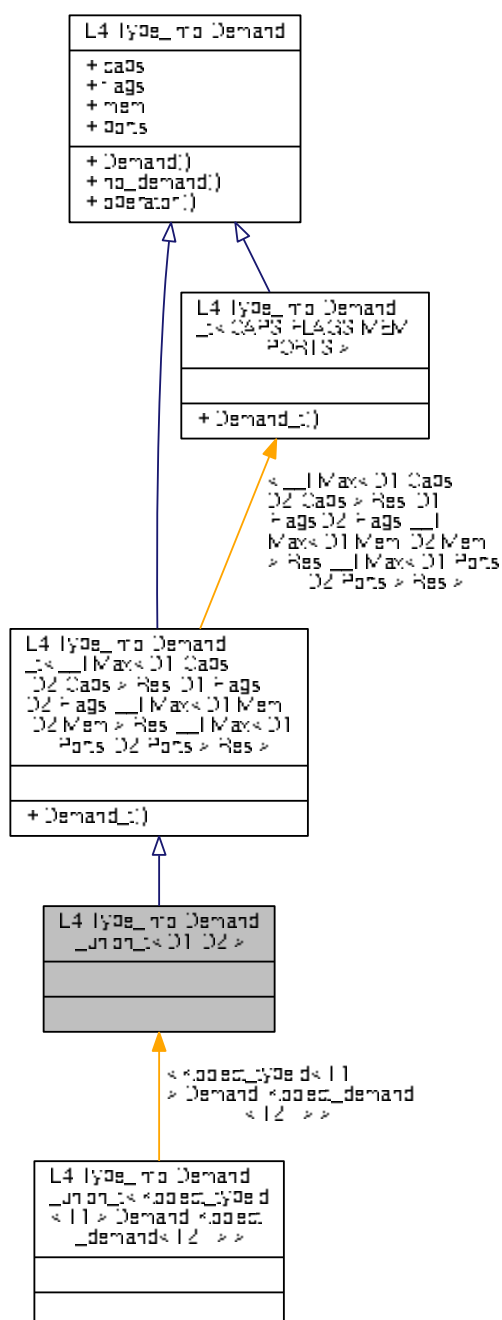
- [l4/sys/__typeinfo.h](#)

14.178 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference

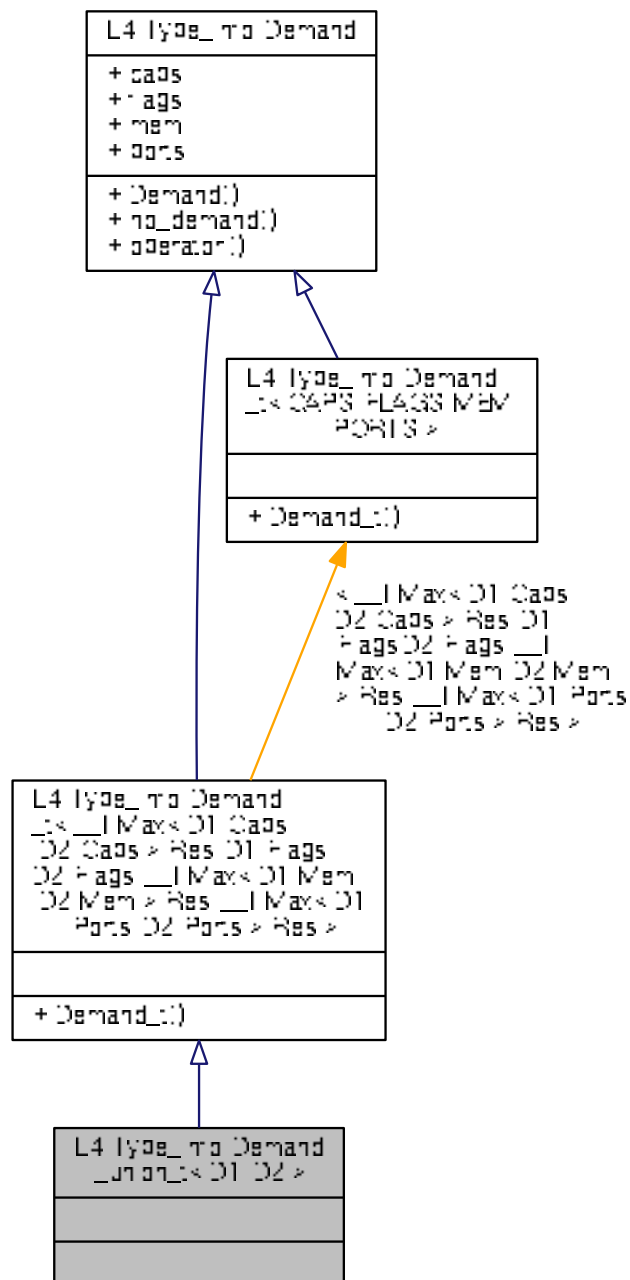
Template type statically describing the combination of two [Demand](#) object.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand_union_t< D1, D2 >:



Collaboration diagram for L4::Type_info::Demand_union_t< D1, D2 >:



Additional Inherited Members

14.178.1 Detailed Description

```
template<typename D1, typename D2>
struct L4::Type_info::Demand_union_t< D1, D2 >
```

Template type statically describing the combination of two [Demand](#) object.

Template Parameters

$D1$	first demand object.
$D2$	second demand object.

Definition at line 583 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

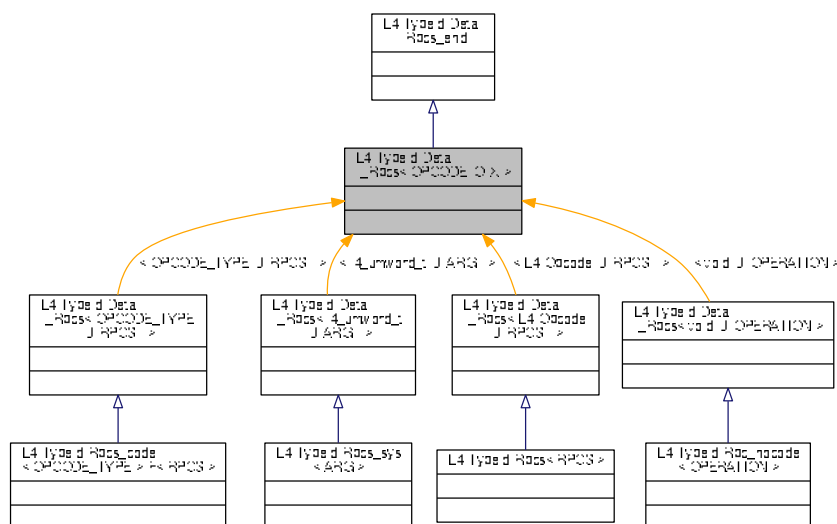
- `l4/sys/__typeinfo.h`

14.179 L4::Typeid::Detail:: Rpcs< OPCODE, O, X > Struct Template Reference

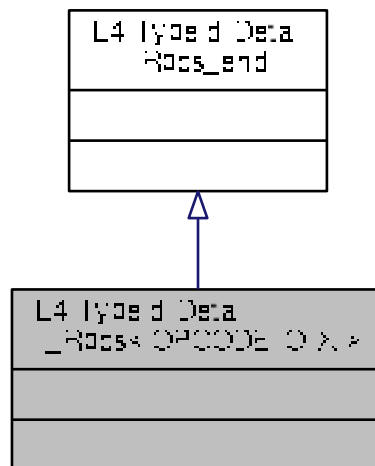
Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc< OPCODE, O, X >:



Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



14.179.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, X >
```

Empty list of RPCs.

Definition at line 375 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

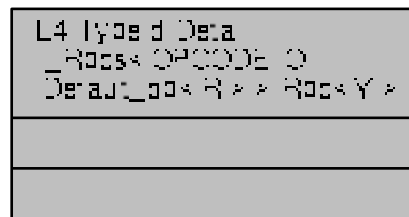
14.180 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >:



14.180.1 Detailed Description

```

template<typename OPCODE, unsigned O, typename R>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >
  
```

Find the given RPC in the list.

Definition at line 409 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

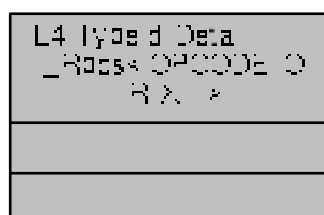
- [l4/sys/__typeinfo.h](#)

14.181 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference

Non-empty list of RPCs.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >:



Data Structures

- struct [Rpc](#)

Find the given RPC in the list.

Public Types

- enum
The opcode value to use for this RPC, may be bogus if the opcode_type is void.
- typedef [_Rpcs](#) type
The list element itself.
- typedef OPCODE [opcode_type](#)
The data type for the opcode.
- typedef R [rpc](#)
The RPC type `L4::lpc::Msg::Rpc_call` or `L4::lpc::Msg::Rpc_inline_call`.
- typedef [_Rpcs](#)< OPCODE, _Get_opcode< R, O >::value+1, X... >::type next
The next RPC in the list or [Rpc_end](#) if this is the last.

14.181.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
struct L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >
```

Non-empty list of RPCs.

Definition at line 379 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

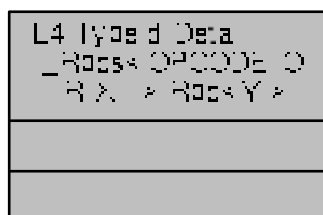
14.182 L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits `L4::Typeid::Detail::Rpc< OP, RPCS >`.

Collaboration diagram for `L4::Typeid::Detail::_Rpcs< OPCODE, O, R, X... >::Rpc< Y >`:



14.182.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >
```

Find the given RPC in the list.

Definition at line 392 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

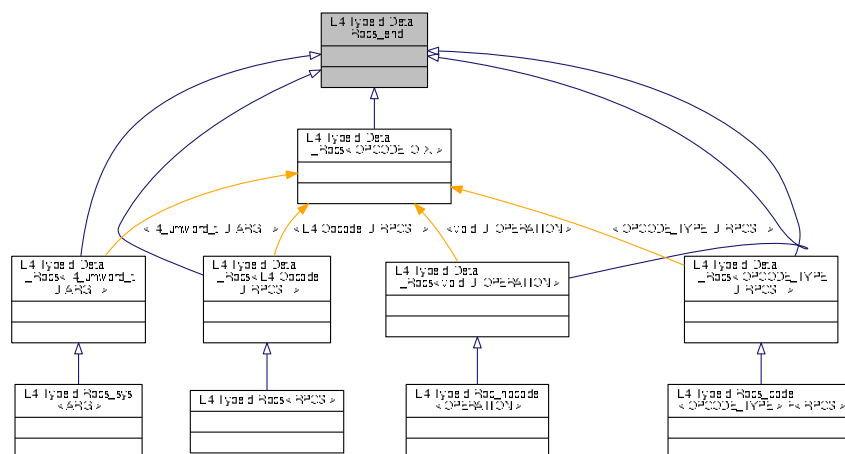
- [l4/sys/__typeinfo.h](#)

14.183 L4::Typeid::Detail::Rpc_end Struct Reference

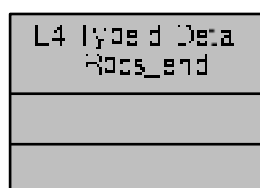
Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc_end:



Collaboration diagram for L4::Typeid::Detail::Rpc_end:



14.183.1 Detailed Description

Internal end-of-list marker.

Definition at line 327 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

14.184 L4::Typeid::P_dispatch< LIST > Struct Template Reference

Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::_P_dispatch< LIST >.

Collaboration diagram for L4::Typeid::P_dispatch< LIST >:



14.184.1 Detailed Description

```
template<typename LIST>
struct L4::Typeid::P_dispatch< LIST >
```

Use for protocol based dispatch stage.

Definition at line 318 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

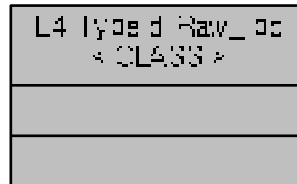
- [l4/sys/__typeinfo.h](#)

14.185 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference

RPCs list for passing raw incoming IPC to the server object.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Raw_ipc< CLASS >:



14.185.1 Detailed Description

```
template<typename CLASS>
struct L4::Typeid::Raw_ipc< CLASS >
```

RPCs list for passing raw incoming IPC to the server object.

Template Parameters

<i>CLASS</i>	The type of the interface (e.g., L4::lcu)
--------------	--

This template allows to have fully handcrafted IPC protocols.

Definition at line [422](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

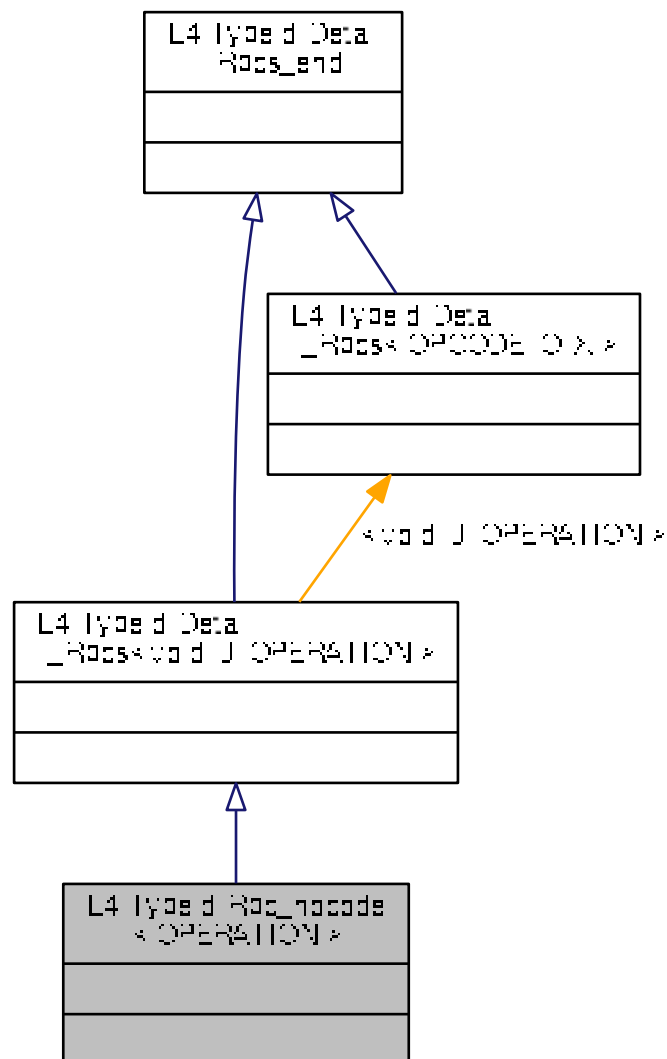
- [l4/sys/__typeinfo.h](#)

14.186 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference

List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```


Collaboration diagram for L4::Typeid::Rpc_nocode< OPERATION >:



14.186.1 Detailed Description

```
template<typename OPERATION>
struct L4::Typeid::Rpc_nocode< OPERATION >
```

List of RPCs of an interface using a single operation without an opcode.

Template Parameters

<code>OPERATION</code>	The RPC operation as defined by L4_RPC etc.
------------------------	---

Definition at line 464 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

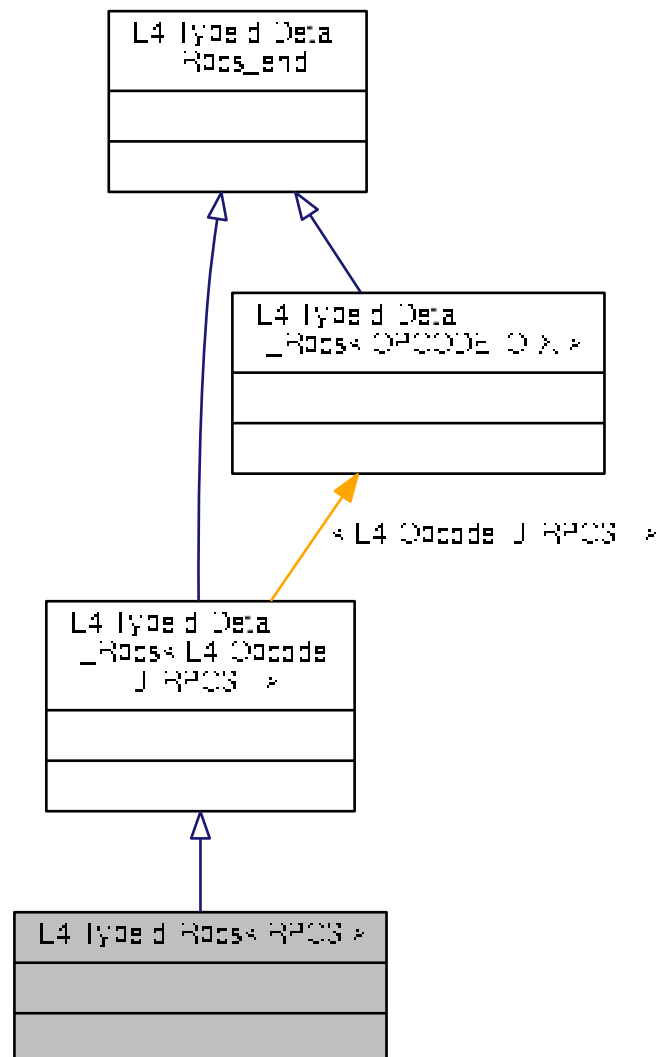
- [l4/sys/__typeinfo.h](#)

14.187 L4::Typeid::Rpc< RPCS > Struct Template Reference

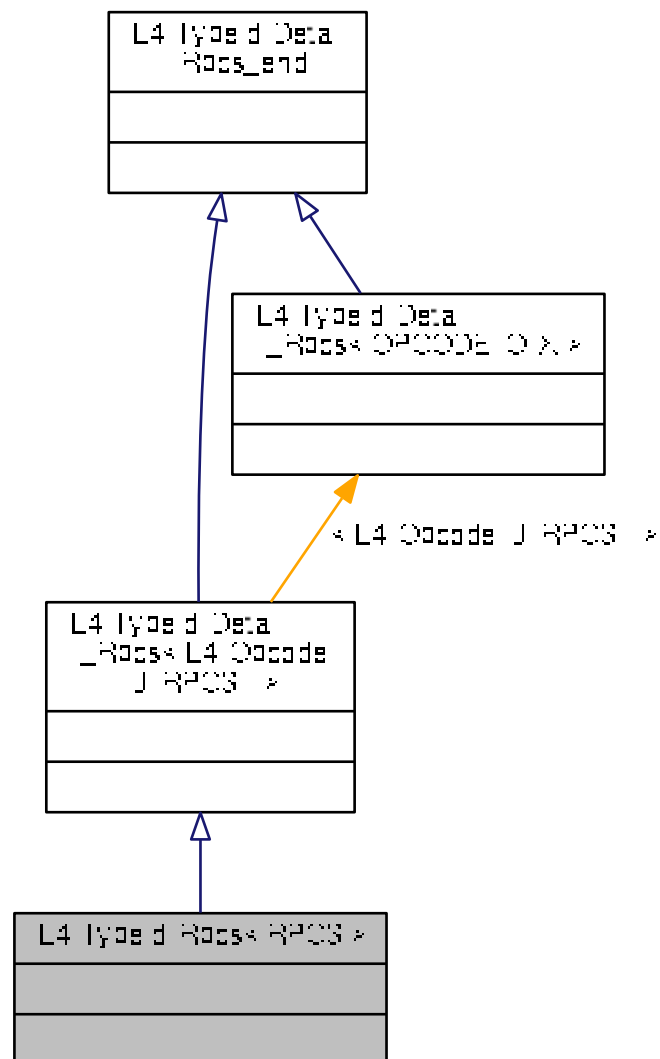
Standard list of RPCs of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc< RPCS >:



Collaboration diagram for L4::Typeid::Rpc< RPCS >:



14.187.1 Detailed Description

```
template<typename ... RPCS>
struct L4::Typeid::Rpc< RPCS >
```

Standard list of RPCs of an interface.

Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---

This is the default list for RPC functions of an interface, it uses [L4::Opcode](#) as opcode type and uses opcodes starting from 0.

Definition at line [438](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

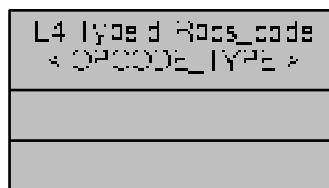
- [l4/sys/__typeinfo.h](#)

14.188 L4::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference

List of RPCs of an interface using a special opcode type.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >:



Data Structures

- struct [F](#)

14.188.1 Detailed Description

```
template<typename OPCODE_TYPE>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >
```

List of RPCs of an interface using a special opcode type.

Template Parameters

<i>OPCODE_TYPE</i>	The data type of the opcode.
--------------------	------------------------------

List for RPC functions of an interface, using OPCODE_TYPE as data type for the opcode, opcodes starting from 0.

Definition at line [449](#) of file [__typeinfo.h](#).

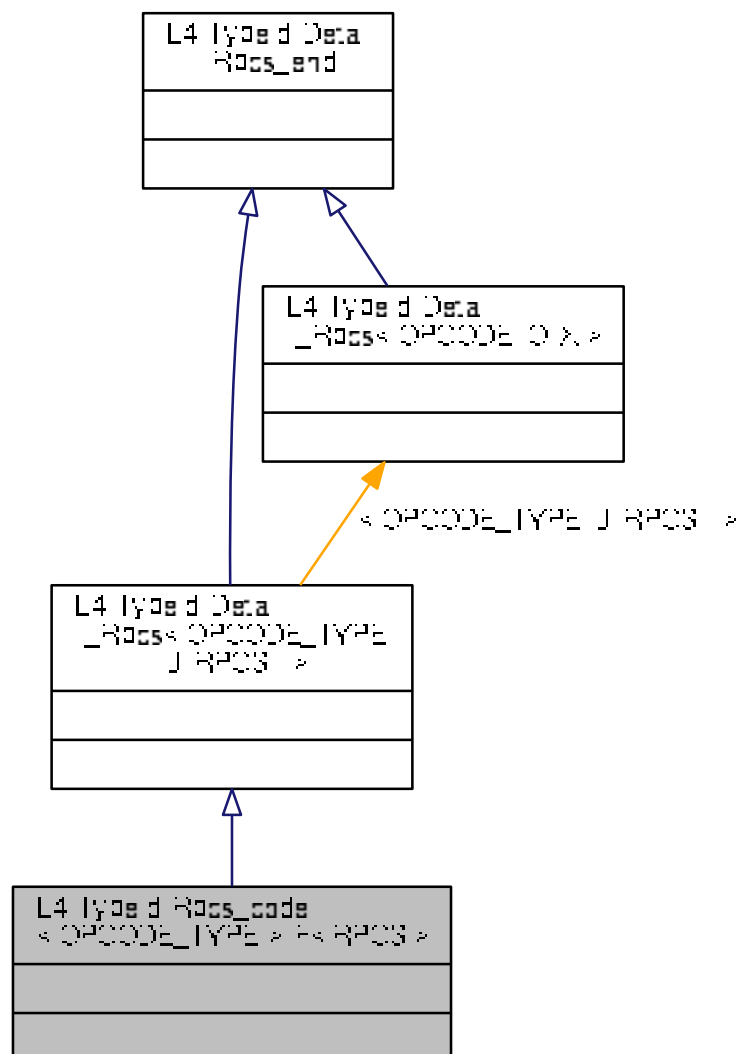
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

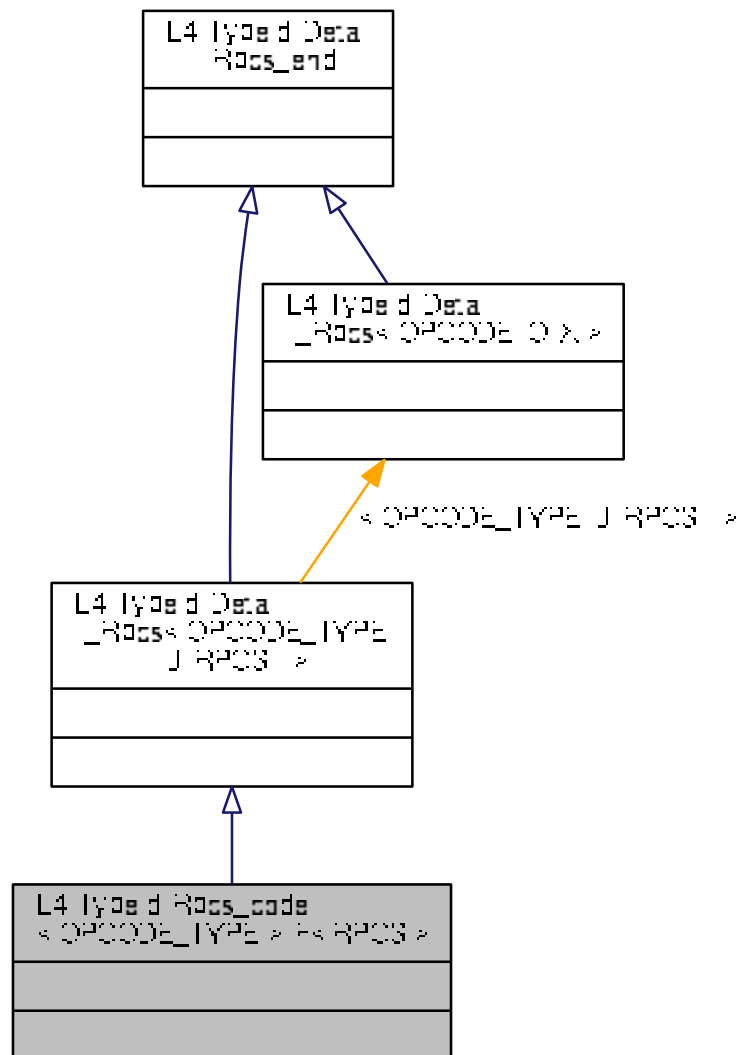
14.189 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >:



Collaboration diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >:



14.189.1 Detailed Description

```

template<typename OPCODE_TYPE>
template<typename ... RPCS>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >

```

Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---

Definition at line 455 of file `__typeinfo.h`.

The documentation for this struct was generated from the following file:

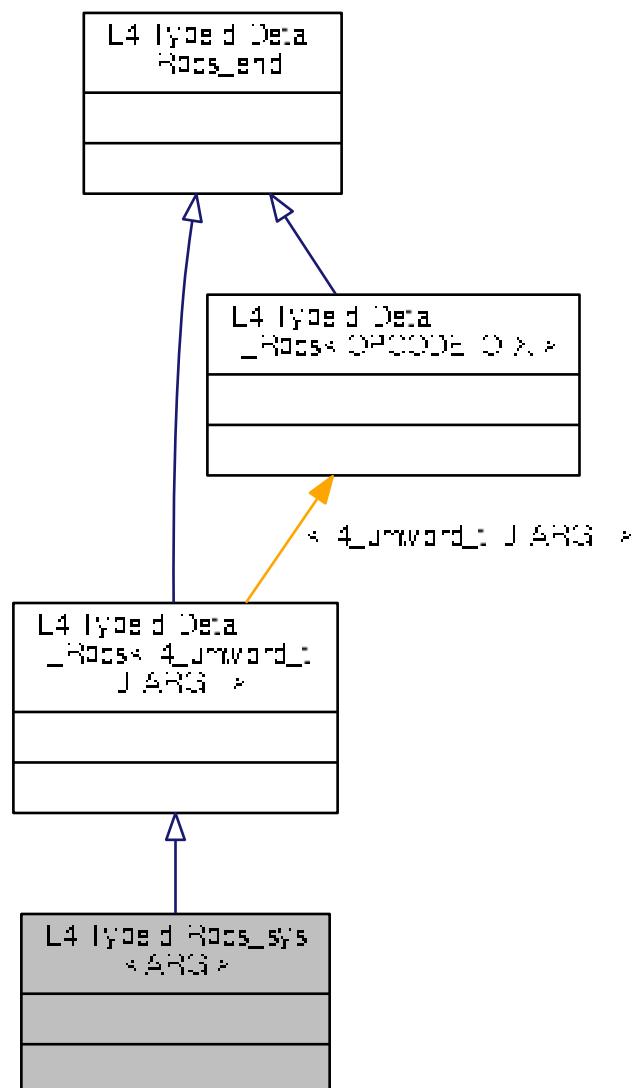
- [l4/sys/__typeinfo.h](#)

14.190 L4::Typeid::Rpcsys< ARG > Struct Template Reference

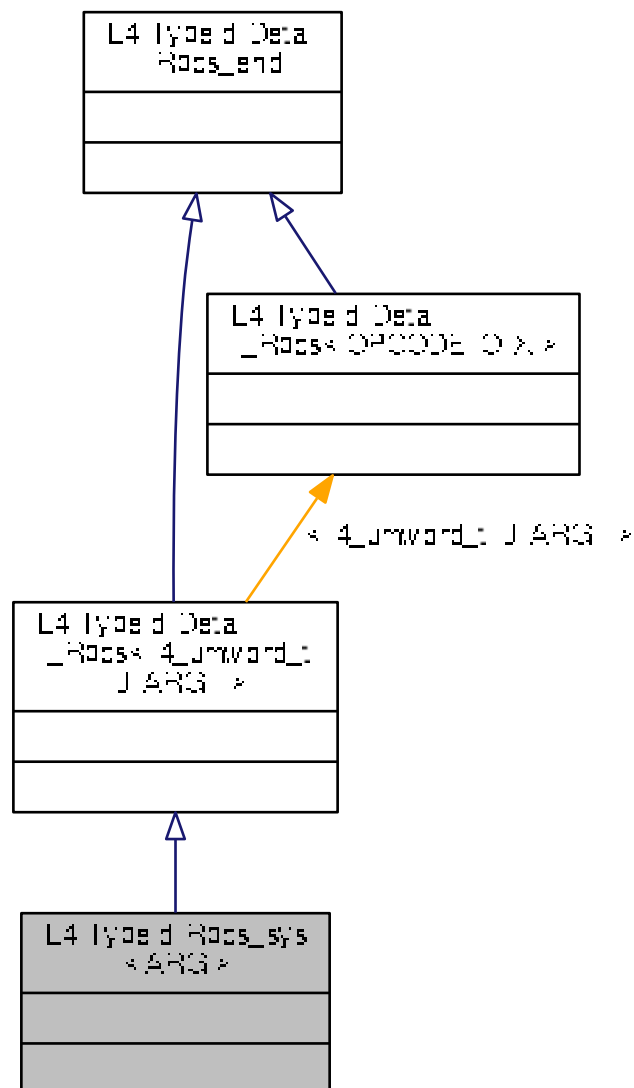
List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcsys< ARG >:



Collaboration diagram for L4::Typeid::Rpcsys< ARG >:



14.190.1 Detailed Description

```
template<typename ... ARG>
struct L4::Typeid::Rpcsys< ARG >
```

List of RPCs typically used for kernel interfaces.

Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---

This list of RPC functions uses `l4_umword_t` as type for the opcode as most kernel protocol do.

Definition at line 475 of file __typeinfo.h.

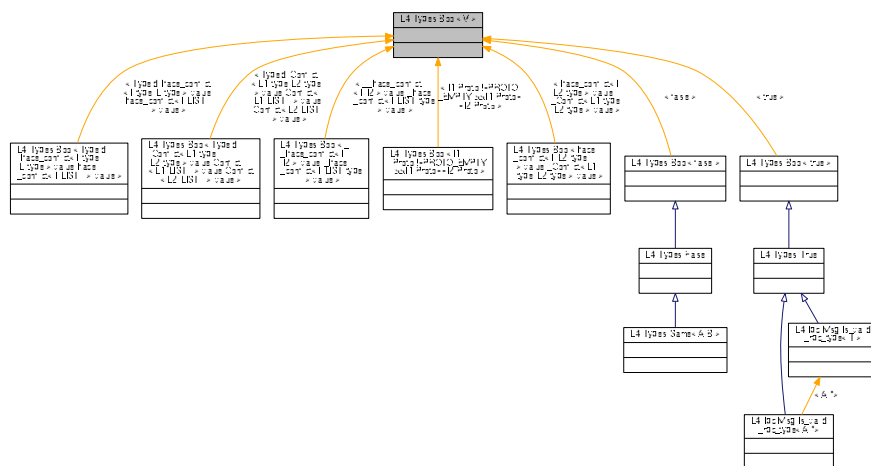
The documentation for this struct was generated from the following file:

- `l4/sys/__typeinfo.h`

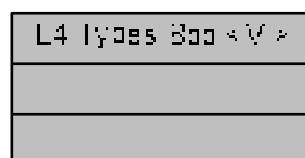
14.191 L4::Types::Bool< V > Struct Template Reference

Boolean meta type.

Inheritance diagram for L4::Types::Bool< V >:



Collaboration diagram for L4::Types::Bool< V >:



Public Types

- `typedef Bool< V > type`
The meta type itself.

14.191.1 Detailed Description

```
template<bool V>  
struct L4::Types::Bool< V >
```

Boolean meta type.

Template Parameters

<i>V</i>	The boolean value
----------	-------------------

Definition at line [165](#) of file [types](#).

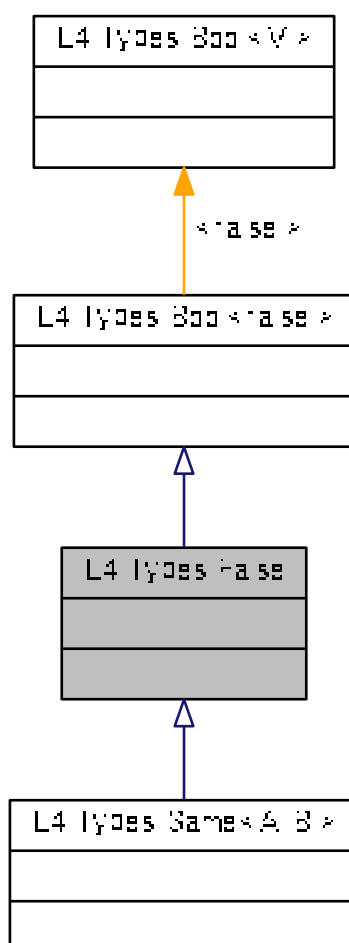
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

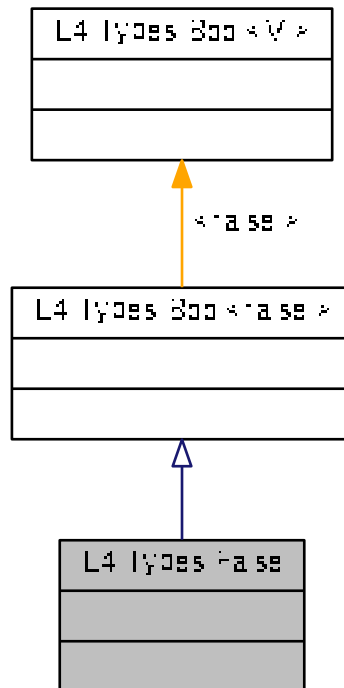
14.192 L4::Types::False Struct Reference

[False](#) meta value.

Inheritance diagram for L4::Types::False:



Collaboration diagram for L4::Types::False:



Additional Inherited Members

14.192.1 Detailed Description

[False](#) meta value.

Definition at line [173](#) of file [types](#).

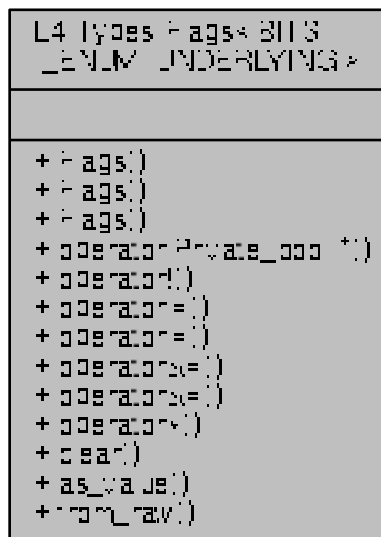
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

14.193 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference

Template for defining typical [Flags](#) bitmaps.

Collaboration diagram for L4::Types::Flags< BITS_ENUM, UNDERLYING >:



Public Types

- enum [None_type](#) { [None](#) }
The none type to get an empty bitmap.
- typedef UNDERLYING [value_type](#)
type of the underlying value
- typedef BITS_ENUM [bits_enum_type](#)
enum type defining a name for each bit
- typedef [Flags](#)< BITS_ENUM, UNDERLYING > [type](#)
the Flags<> type itself

Public Member Functions

- [Flags](#) ([None_type](#))
Make an empty bitmap.
- [Flags](#) ()
Make default [Flags](#).
- [Flags](#) (BITS_ENUM e)
Make flags from bit name.
- [operator Private_bool *](#) () const
Support for `if (flags)` syntax (test for non-empty flags).
- [bool operator!](#) () const
Support for `if (!flags)` syntax (test for empty flags).
- [type & operator|=](#) ([type](#) rhs)
Support `|=` of two compatible [Flags](#) types.

- `type & operator| = (bits_enum_type rhs)`
Support `| =` of *Flags* type and bit name.
- `type & operator& = (type rhs)`
Support `& =` of two compatible *Flags* types.
- `type & operator& = (bits_enum_type rhs)`
Support `& =` of *Flags* type and bit name.
- `type operator~ () const`
Support `~` for *Flags* types.
- `type & clear (bits_enum_type flag)`
Clear the given flag.
- `value_type as_value () const`
Get the underlying value.

Static Public Member Functions

- static `type from_raw (value_type v)`
Make flags from a raw value of *value_type*.

Friends

- `type operator| (type lhs, type rhs)`
Support `|` of two compatible *Flags* types.
- `type operator| (type lhs, bits_enum_type rhs)`
Support `|` of *Flags* type and bit name.
- `type operator& (type lhs, type rhs)`
Support `&` of two compatible *Flags* types.
- `type operator& (type lhs, bits_enum_type rhs)`
Support `&` of *Flags* type and bit name.

14.193.1 Detailed Description

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
class L4::Types::Flags< BITS_ENUM, UNDERLYING >
```

Template for defining typical *Flags* bitmaps.

Template Parameters

<i>BITS_ENUM</i>	enum type that defines a name for each bit in the bitmap. The values of the enum members must be the number of the bit (<i>not</i> a mask).
<i>UNDERLYING</i>	The underlying data type used to represent the bitmap.

The resulting data type provides a type-safe version that allows bitwise `and` and `or` operations with the `BITS_ENUM` members. As well as, test for `0` or `!0`.

Example:

```
enum Test_flag
{
    Do_weak_tests,
    Do_strong_tests
};

typedef L4::Types::Flags<Test_flag> Test_flags;

Test_flags x = Do_weak_tests;

if (x & Do_strong_tests) { ... }
x |= Do_strong_tests;
if (x & Do_strong_tests) { ... }
```

Definition at line 63 of file [types](#).

14.193.2 Member Enumeration Documentation

14.193.2.1 None_type

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
enum L4::Types::Flags::None_type
```

The none type to get an empty bitmap.

Enumerator

None	Use this to get an empty bitmap.
------	----------------------------------

Definition at line 80 of file [types](#).

14.193.3 Constructor & Destructor Documentation

14.193.3.1 Flags() [1/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    None_type ) [inline]
```

Make an empty bitmap.

Usually used for implicit conversion from [Flags::None](#).

```
Flags x = Flags::None;
```

Definition at line 90 of file [types](#).

14.193.3.2 Flags() [2/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    BITS_ENUM e ) [inline]
```

Make flags from bit name.

Usually used for implicit conversion for a bit name.

```
Test_flags f = Do_strong_tests;
```

Definition at line 103 of file [types](#).

14.193.4 Member Function Documentation

14.193.4.1 clear()

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
type& L4::Types::Flags< BITS_ENUM, UNDERLYING >::clear (
    bits_enum_type flag ) [inline]
```

Clear the given flag.

Parameters

<i>flag</i>	The flag that shall be cleared.
-------------	---------------------------------

`flags.clear(The_flag)` is a shortcut for `flags &= ~Flags(The_flag)`.

Definition at line 154 of file [types](#).

References [L4::Types::Flags< BITS_ENUM, UNDERLYING >::operator&=\(\)](#).

Here is the call graph for this function:



14.193.4.2 from_raw()

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
static type L4::Types::Flags< BITS_ENUM, UNDERLYING >::from_raw (
    value_type v ) [inline], [static]
```

Make flags from a raw value of *value_type*.

This function may be used for example in C wrapper code.

Definition at line 110 of file [types](#).

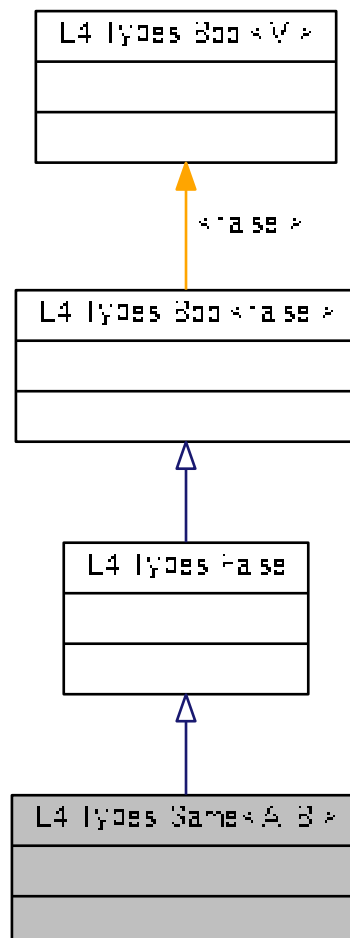
The documentation for this class was generated from the following file:

- [l4/sys/cxx/types](#)

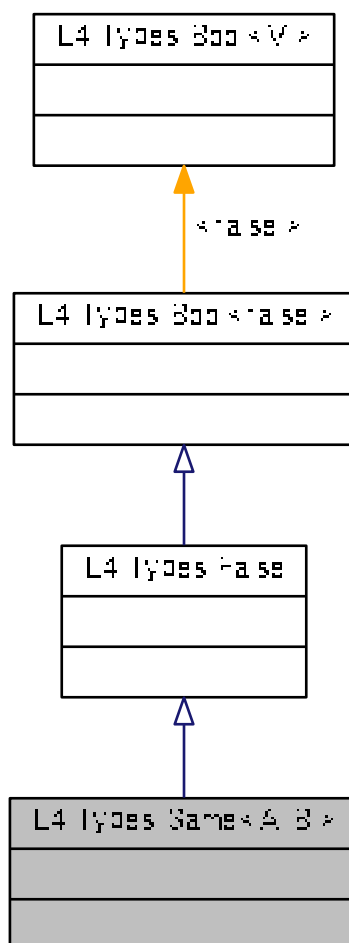
14.194 L4::Types::Same< A, B > Struct Template Reference

Compare two data types for equality.

Inheritance diagram for L4::Types::Same< A, B >:



Collaboration diagram for L4::Types::Same< A, B >:



Additional Inherited Members

14.194.1 Detailed Description

```
template<typename A, typename B>
struct L4::Types::Same< A, B >
```

Compare two data types for equality.

Template Parameters

<i>A</i>	The first data type
<i>B</i>	The second data type

The result is the boolean [True](#) if A and B are the same types.

Definition at line [189](#) of file [types](#).

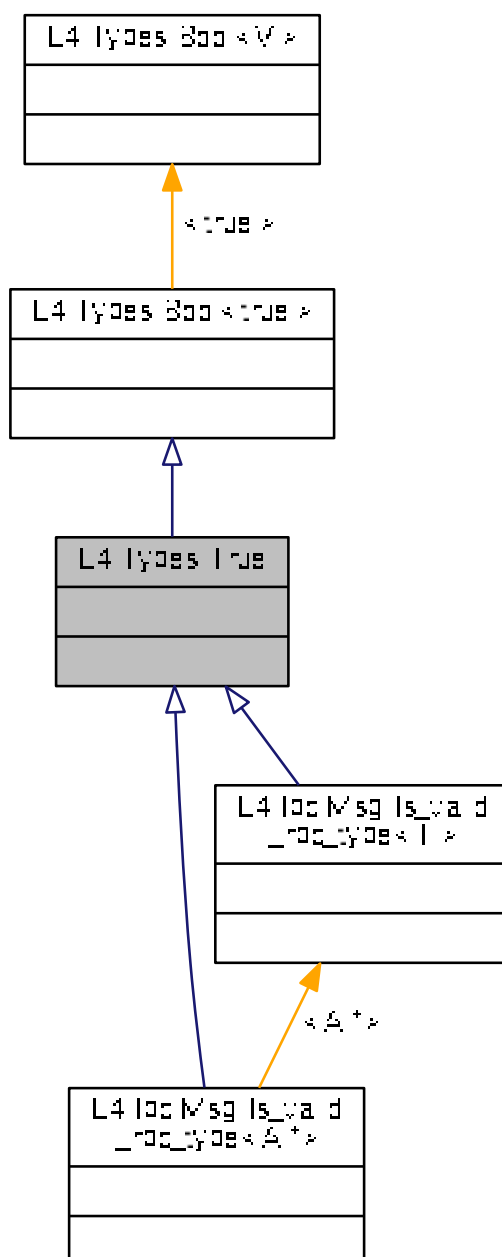
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

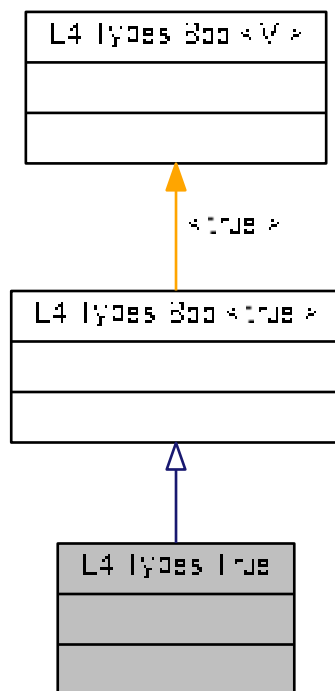
14.195 L4::Types::True Struct Reference

[True](#) meta value.

Inheritance diagram for L4::Types::True:



Collaboration diagram for L4::Types::True:



Additional Inherited Members

14.195.1 Detailed Description

[True](#) meta value.

Definition at line 177 of file [types](#).

The documentation for this struct was generated from the following file:

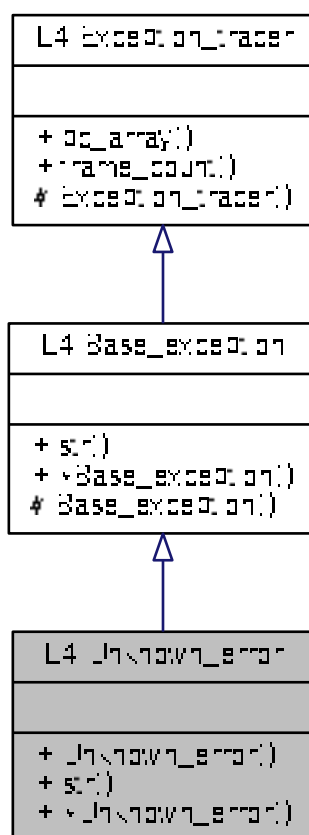
- [l4/sys/cxx/types](#)

14.196 L4::Unknown_error Class Reference

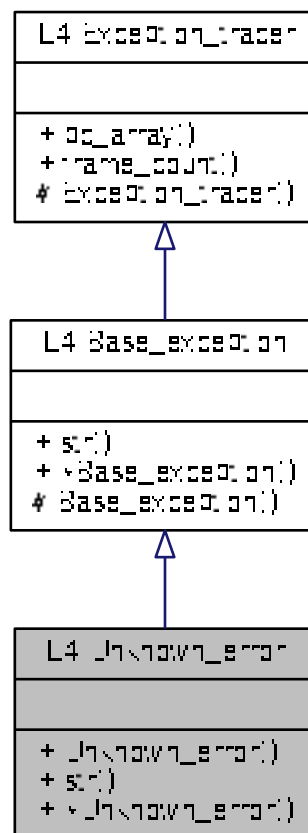
[Exception](#) for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown_error:



Collaboration diagram for L4::Unknown_error:



Public Member Functions

- `char const * str () const throw ()`
Return a human readable string for the exception.

Additional Inherited Members

14.196.1 Detailed Description

Exception for an unknown condition.

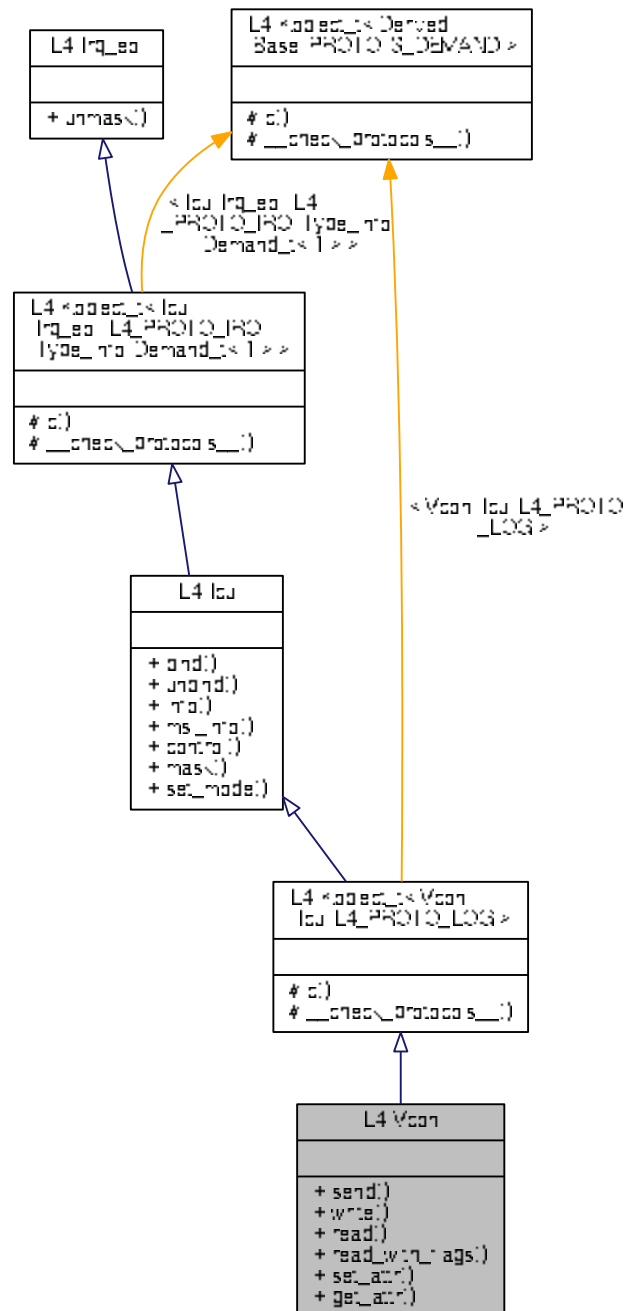
This error is usually used when a server returns an unknown return state to the client, this may indicate incompatible messages used by the client and the server.

Definition at line 219 of file [exceptions](#).

The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

Collaboration diagram for L4::Vcon:



Public Member Functions

- `l4_msgtag_t send` (char const *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const throw ()
Send data to *this* virtual console.
- long `write` (char const *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const throw ()
Write data to *this* virtual console.
- int `read` (char *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const throw ()

Read data from *this* virtual console.

- `int read_with_flags (char *buf, unsigned size, l4_utcb_t *utcb=l4_utcb()) const throw ()`

Read data from *this* virtual console which also returns flags.

- `l4_msgtag_t set_attr (l4_vcon_attr_t const *attr, l4_utcb_t *utcb=l4_utcb()) const throw ()`

Set the attributes of *this* virtual console.

- `l4_msgtag_t get_attr (l4_vcon_attr_t *attr, l4_utcb_t *utcb=l4_utcb()) const throw ()`

Get attributes of *this* virtual console.

Additional Inherited Members

14.197.1 Detailed Description

C++ [L4 Vcon](#) interface.

[L4::Vcon](#) is a virtual console for simple character-based input and output. The interrupt for read events is provided by the virtual key interrupt.

Include File

```
#include <l4/sys/vcon>
```

See the [Virtual Console](#) for the C interface.

Definition at line [43](#) of file [vcon](#).

14.197.2 Member Function Documentation

14.197.2.1 get_attr()

```
l4_msgtag_t L4::Vcon::get_attr (
    l4_vcon_attr_t * attr,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Get attributes of *this* virtual console.

Parameters

out	<i>attr</i>	Attribute structure. Contains the attributes after a successful call of this function.
	<i>utcb</i>	UTCB pointer of the calling thread.

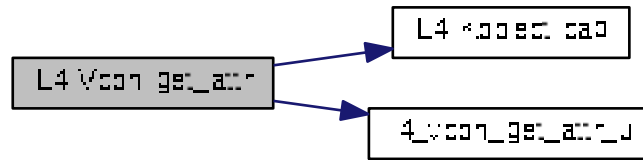
Returns

Syscall return tag.

Definition at line [145](#) of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



14.197.2.2 read()

```
int L4::Vcon::read (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Read data from this virtual console.

Parameters

out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB pointer of the calling thread.

Return values

<0	Error code.
> <i>size</i>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<= <i>size</i>	Number of bytes read.

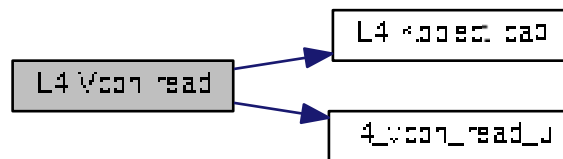
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line 94 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:



14.197.2.3 read_with_flags()

```

int L4::Vcon::read_with_flags (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
  
```

Read data from this virtual console which also returns flags.

Parameters

out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB pointer of the calling thread.

Return values

<0	Error code.
> <i>size</i>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<= <i>size</i>	Number of bytes read.

If this function returns a positive value the caller can check the [L4_VCON_READ_STAT_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4_VCON_READ_SIZE_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line 119 of file [vcon](#).

14.197.2.4 send()

```
l4_msgtag_t L4::Vcon::send (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Send data to this virtual console.

Parameters

<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTBC pointer of the calling thread.

Returns

Syscall return tag

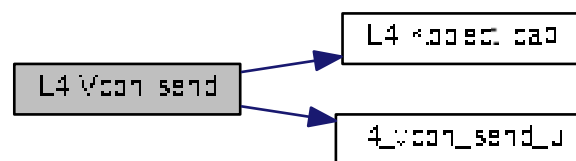
Note

Size must not exceed [L4_VCON_WRITE_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for send errors, do not use [l4_error\(\)](#), as [l4_error\(\)](#) will always return an error.

Definition at line 63 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:



14.197.2.5 set_attr()

```
l4_msgtag_t L4::Vcon::set_attr (
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Set the attributes of this virtual console.

Parameters

<i>attr</i>	Attribute structure with the attributes for the virtual console.
<i>utcb</i>	UTCB pointer of the calling thread.

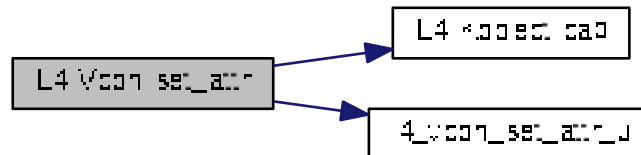
Returns

Syscall return tag.

Definition at line 132 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



14.197.2.6 write()

```

long L4::Vcon::write (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw()    [inline]

```

Write data to this virtual console.

Parameters

<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB pointer of the calling thread.

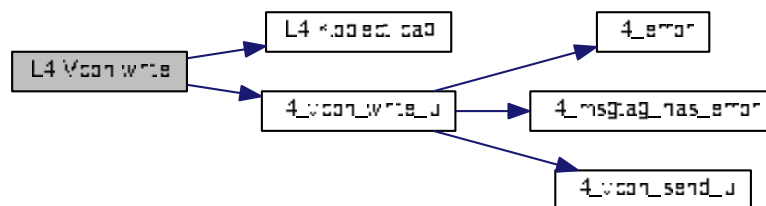
Return values

<0	Error.
>=0	Number of bytes written to the virtual console.

Definition at line 77 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



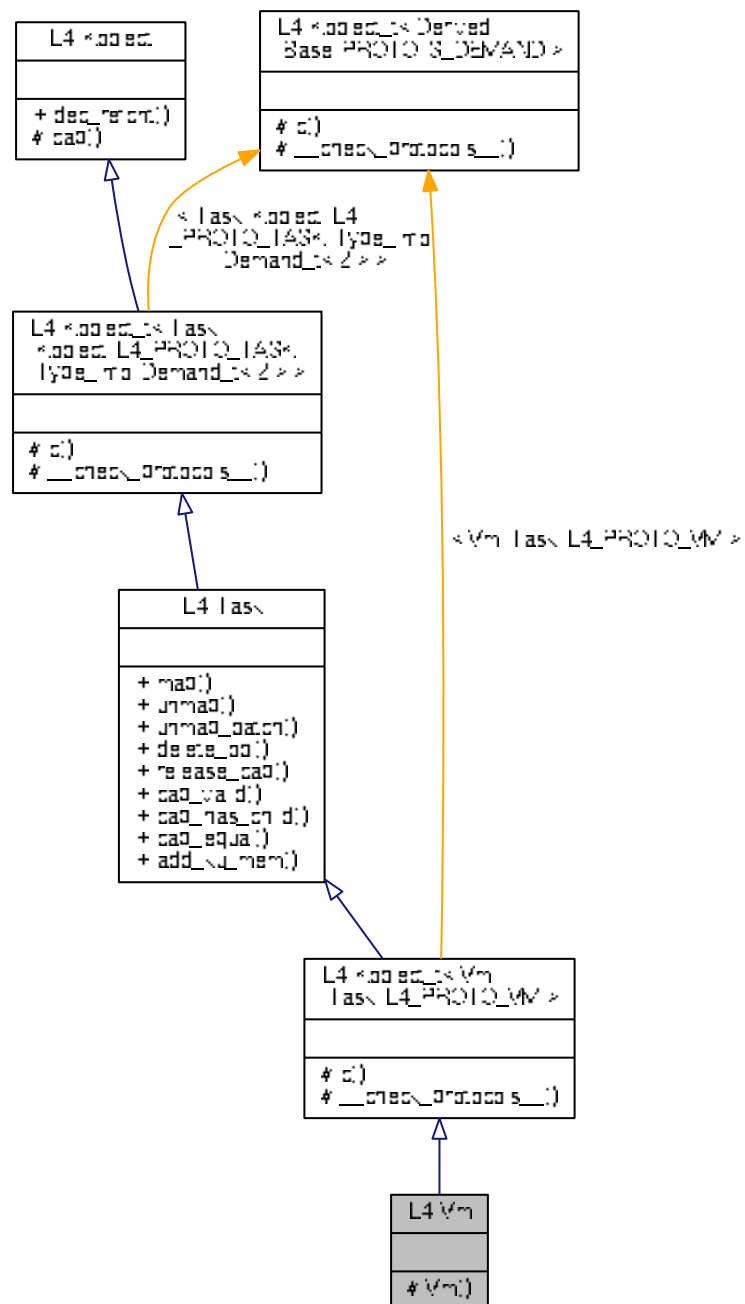
The documentation for this class was generated from the following file:

- [l4/sys/vcon](#)

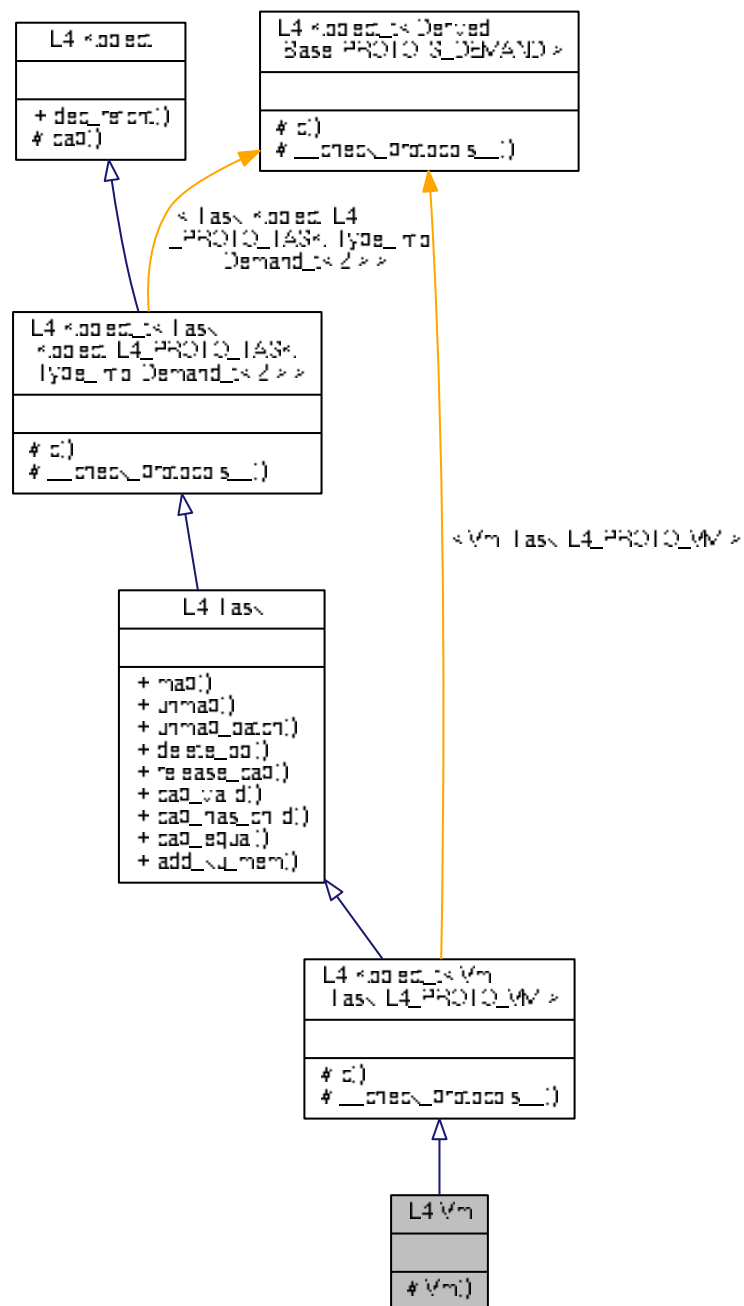
14.198 L4::Vm Class Reference

Virtual machine.

Inheritance diagram for L4::Vm:



Collaboration diagram for L4::Vm:



Additional Inherited Members

14.198.1 Detailed Description

Virtual machine.

[L4::Vm](#) is a specialisation of [L4::Task](#), used for virtual machines. The microkernel employs an appropriate page-table format for hosting VMs, such as ePT on VT-x.

Definition at line [40](#) of file [vm](#).

The documentation for this class was generated from the following file:

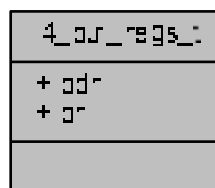
- [l4/sys/vm](#)

14.199 l4_buf_regs_t Struct Reference

Encapsulation of the buffer-registers block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for `l4_buf_regs_t`:



Data Fields

- [l4_umword_t](#) `bdr`
Buffer descriptor.
- [l4_umword_t](#) `br` [[L4_UTCB_GENERIC_BUFFERS_SIZE](#)]
Buffer registers.

14.199.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line [93](#) of file [utcb.h](#).

The documentation for this struct was generated from the following file:

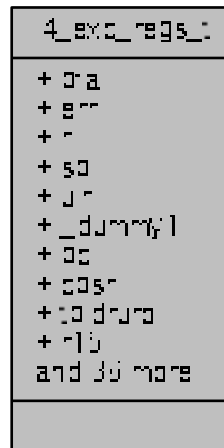
- [l4/sys/utcb.h](#)

14.200 l4_exc_regs_t Struct Reference

UTCB structure for exceptions.

```
#include <utcb.h>
```

Collaboration diagram for l4_exc_regs_t:



Data Fields

- [l4_umword_t pfa](#)
page fault address
- [l4_umword_t err](#)
error code
- [l4_umword_t r \[13\]](#)
registers
- [l4_umword_t sp](#)
stack pointer
- [l4_umword_t ulr](#)
ulr
- [l4_umword_t _dummy1](#)
dummy
- [l4_umword_t pc](#)
pc
- [l4_umword_t cpsr](#)
cpsr
- [l4_umword_t tpidrro](#)
Thread-ID register.
- [l4_umword_t r15](#)
r15

- [l4_umword_t r14](#)
r14
- [l4_umword_t r13](#)
r13
- [l4_umword_t r12](#)
r12
- [l4_umword_t r11](#)
r11
- [l4_umword_t r10](#)
r10
- [l4_umword_t r9](#)
r9
- [l4_umword_t r8](#)
r8
- [l4_umword_t rdi](#)
rdi
- [l4_umword_t rsi](#)
rsi
- [l4_umword_t rbp](#)
rbp
- [l4_umword_t rbx](#)
rbx
- [l4_umword_t rdx](#)
rdx
- [l4_umword_t rcx](#)
rcx
- [l4_umword_t rax](#)
rax
- [l4_umword_t trapno](#)
trap number
- [l4_umword_t ip](#)
instruction pointer
- [l4_umword_t dummy1](#)
dummy
- [l4_umword_t flags](#)
rflags
- [l4_umword_t ss](#)
stack segment register
- [l4_umword_t es](#)
es register
- [l4_umword_t ds](#)
ds register
- [l4_umword_t gs](#)
gs register
- [l4_umword_t fs](#)
fs register
- [l4_umword_t edi](#)
edi register
- [l4_umword_t esi](#)
esi register
- [l4_umword_t ebp](#)

- ebp register*
- [l4_umword_t ebx](#)
ebx register
- [l4_umword_t edx](#)
edx register
- [l4_umword_t ecx](#)
ecx register
- [l4_umword_t eax](#)
eax register

14.200.1 Detailed Description

UTCB structure for exceptions.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 38 of file [utcb.h](#).

14.200.2 Field Documentation

14.200.2.1 flags

[l4_umword_t](#) l4_exc_regs_t::flags

rflags

eflags

Definition at line 80 of file [utcb.h](#).

14.200.2.2 ss

[l4_umword_t](#) l4_exc_regs_t::ss

stack segment register

ss register

Definition at line 82 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

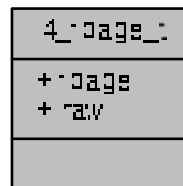
- [arm/l4/sys/utcb.h](#)

14.201 l4_fpage_t Union Reference

L4 flexpage type.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4_fpage_t:



Data Fields

- [l4_umword_t fpage](#)
Raw value.
- [l4_umword_t raw](#)
Raw value.

14.201.1 Detailed Description

L4 flexpage type.

Definition at line 81 of file [__l4_fpage.h](#).

The documentation for this union was generated from the following file:

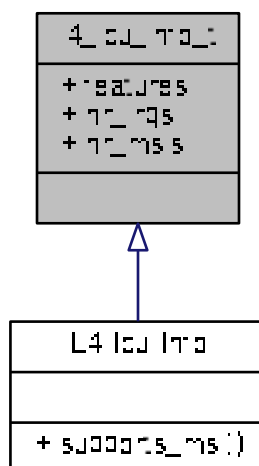
- l4/sys/__l4_fpage.h

14.202 l4_icu_info_t Struct Reference

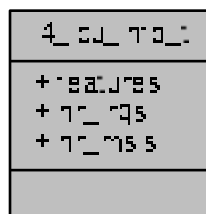
Info structure for an ICU.

```
#include <icu.h>
```

Inheritance diagram for l4_icu_info_t:



Collaboration diagram for l4_icu_info_t:



Data Fields

- unsigned [features](#)

Feature flags.

- unsigned [nr_irqs](#)

The number of IRQ lines supported by the ICU,.

- unsigned [nr_msis](#)

The number of MSI vectors supported by the ICU,.

14.202.1 Detailed Description

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

[l4_icu_info\(\)](#).

Definition at line 159 of file [icu.h](#).

14.202.2 Field Documentation

14.202.2.1 features

```
unsigned l4_icu_info_t::features
```

Feature flags.

If [L4_ICU_FLAG_MSI](#) is set the ICU supports MSIs.

Definition at line 166 of file [icu.h](#).

The documentation for this struct was generated from the following file:

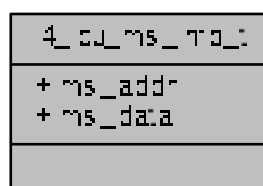
- [l4/sys/icu.h](#)

14.203 l4_icu_msi_info_t Struct Reference

Info to use for a specific MSI.

```
#include <icu.h>
```

Collaboration diagram for `l4_icu_msi_info_t`:



Data Fields

- [l4_uint64_t msi_addr](#)
Value to use as address when sending this MSI.
- [l4_uint32_t msi_data](#)
Value to use as data written to msi_addr, when sending this MSI.

14.203.1 Detailed Description

Info to use for a specific MSI.

Definition at line 180 of file [icu.h](#).

14.203.2 Field Documentation

14.203.2.1 msi_data

```
l4\_uint32\_t l4_icu_msi_info_t::msi_data
```

Value to use as data written to msi_addr, when sending this MSI.

Definition at line 185 of file [icu.h](#).

The documentation for this struct was generated from the following file:

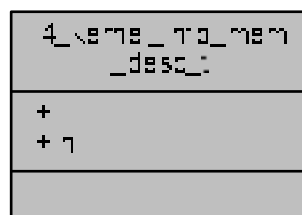
- [l4/sys/icu.h](#)

14.204 l4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for `l4_kernel_info_mem_desc_t`:



14.204.1 Detailed Description

Memory descriptor data structure.

Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

Definition at line 74 of file [memdesc.h](#).

The documentation for this struct was generated from the following file:

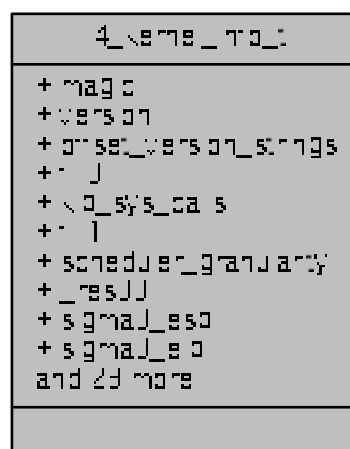
- [l4/sys/memdesc.h](#)

14.205 l4_kernel_info_t Struct Reference

[L4 Kernel Interface Page](#).

```
#include <__kip-32bit.h>
```

Collaboration diagram for `l4_kernel_info_t`:



Data Fields

- [l4_uint32_t magic](#)
Kernel Info Page identifier ("L4μK").
- [l4_uint32_t version](#)
Kernel version.
- [l4_uint8_t offset_version_strings](#)
offset to version string
- [l4_uint8_t fill0](#) [3]
reserved
- [l4_uint8_t kip_sys_calls](#)
pointer to system calls
- [l4_uint8_t fill1](#) [3]
reserved
- [l4_umword_t scheduler_granularity](#)
for rounding time slices
- [l4_umword_t _res00](#) [3]
default_kdebug_end
- [l4_umword_t sigma0_esp](#)
Sigma0 start stack pointer.
- [l4_umword_t sigma0_eip](#)
Sigma0 instruction pointer.
- [l4_umword_t _res01](#) [2]
reserved
- [l4_umword_t sigma1_esp](#)
Sigma1 start stack pointer.
- [l4_umword_t sigma1_eip](#)
Sigma1 instruction pointer.
- [l4_umword_t _res02](#) [2]
reserved
- [l4_umword_t root_esp](#)
Root task stack pointer.
- [l4_umword_t root_eip](#)
Root task instruction pointer.
- [l4_umword_t _res03](#) [2]
reserved
- [l4_umword_t _res50](#) [1]
reserved
- [l4_umword_t mem_info](#)
memory information
- [l4_umword_t _res58](#) [2]
reserved
- [l4_umword_t _res04](#) [16]
reserved
- [l4_umword_t _res05](#) [2]
reserved
- [l4_umword_t frequency_cpu](#)
CPU frequency in kHz.
- [l4_umword_t frequency_bus](#)
Bus frequency.
- [l4_umword_t _res06](#) [10]

- reserved*
- [l4_umword_t user_ptr](#)
user_ptr
- [l4_umword_t vhw_offset](#)
offset to vhw structure
- [l4_uint64_t magic](#)
Kernel Info Page identifier ("L4μK").
- [l4_uint64_t version](#)
Kernel version.
- [l4_uint8_t fill2](#) [7]
reserved
- [l4_uint8_t fill3](#) [7]
reserved
- [l4_umword_t _res_a0](#) [1]
reserved
- [l4_umword_t _res_b0](#) [2]
reserver

14.205.1 Detailed Description

[L4 Kernel Interface Page](#).

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 38 of file [__kip-32bit.h](#).

The documentation for this struct was generated from the following files:

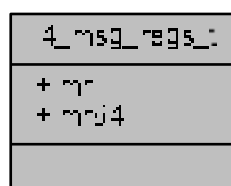
- [l4/sys/__kip-32bit.h](#)
- [l4/sys/__kip-64bit.h](#)

14.206 l4_msg_regs_t Union Reference

Encapsulation of the message-register block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for `l4_msg_regs_t`:



Data Fields

- [l4_umword_t mr](#) [[L4_UTCB_GENERIC_DATA_SIZE](#)]
Message registers.
- [l4_uint64_t mr64](#) [[L4_UTCB_GENERIC_DATA_SIZE](#)/(sizeof([l4_uint64_t](#))/sizeof([l4_umword_t](#)))]
Message registers 64bit alias.

14.206.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

Examples:

[examples/sys/utcb-ipc/main.c](#).

Definition at line 78 of file [utcb.h](#).

The documentation for this union was generated from the following file:

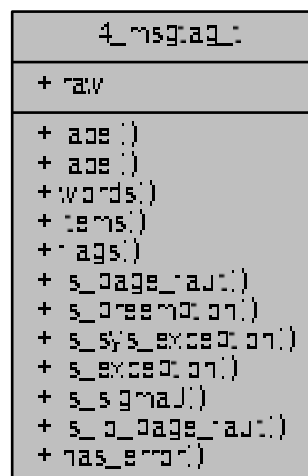
- [l4/sys/utcb.h](#)

14.207 l4_msgtag_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

Collaboration diagram for l4_msgtag_t:



Public Member Functions

- long [label](#) () const throw ()
Get the protocol value.
- void [label](#) (long v) throw ()
Set the protocol value.
- unsigned [words](#) () const throw ()
Get the number of untyped words.
- unsigned [items](#) () const throw ()
Get the number of typed items.
- unsigned [flags](#) () const throw ()
Get the flags value.
- bool [is_page_fault](#) () const throw ()
Test if protocol indicates page-fault protocol.
- bool [is_preemption](#) () const throw ()
Test if protocol indicates preemption protocol.
- bool [is_sys_exception](#) () const throw ()
Test if protocol indicates system-exception protocol.
- bool [is_exception](#) () const throw ()
Test if protocol indicates exception protocol.
- bool [is_sigma0](#) () const throw ()
Test if protocol indicates sigma0 protocol.
- bool [is_io_page_fault](#) () const throw ()
Test if protocol indicates IO-page-fault protocol.
- unsigned [has_error](#) () const throw ()
Test if flags indicate an error.

Data Fields

- [l4_mword_t](#) raw
raw value

14.207.1 Detailed Description

Message tag data structure.

Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

Examples:

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), [examples/libs/l4re/streammap/server.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 159 of file [types.h](#).

14.207.2 Member Function Documentation

14.207.2.1 flags()

```
unsigned l4_msgtag_t::flags ( ) const throw ( ) [inline]
```

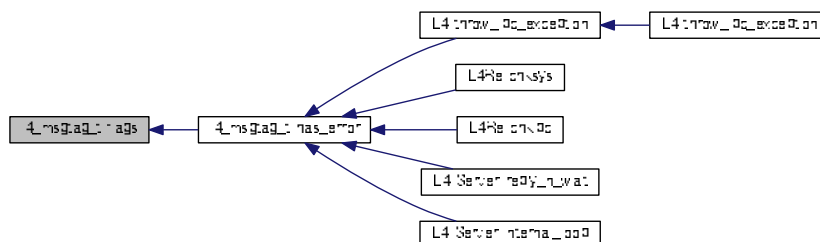
Get the flags value.

The flags are a combination of the flags defined by `l4_msgtag_flags`.

Definition at line 177 of file types.h.

Referenced by [has_error\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

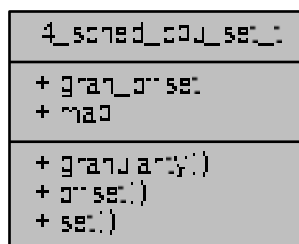
- `l4/sys/types.h`

14.208 l4_sched_cpu_set_t Struct Reference

CPU sets.

```
#include <scheduler.h>
```

Collaboration diagram for l4_sched_cpu_set_t:



Public Member Functions

- unsigned char [granularity](#) () const
- unsigned [offset](#) () const
- void [set](#) (unsigned char [granularity](#), unsigned [offset](#))
Set offset and granularity.

Data Fields

- [l4_umword_t](#) [gran_offset](#)
Combination of granularity and offset.
- [l4_umword_t](#) [map](#)
Bitmap of CPUs.

14.208.1 Detailed Description

CPU sets.

Examples:

[examples/sys/migrate/thread_migrate.cc](#).

Definition at line 44 of file [scheduler.h](#).

14.208.2 Member Function Documentation

14.208.2.1 [granularity\(\)](#)

```
unsigned char l4_sched_cpu_set_t::granularity ( ) const [inline]
```

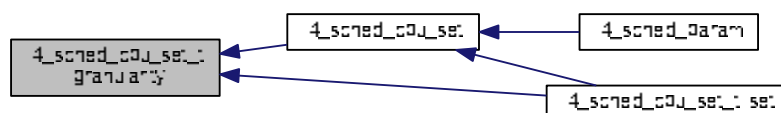
Returns

Get granularity value

Definition at line 67 of file [scheduler.h](#).

Referenced by [l4_sched_cpu_set\(\)](#), and [set\(\)](#).

Here is the caller graph for this function:



14.208.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset ( ) const [inline]
```

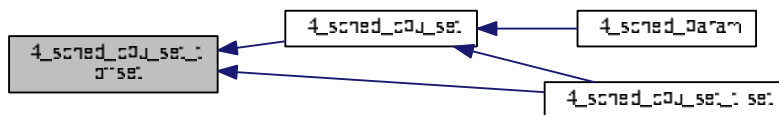
Returns

Get offset value

Definition at line 69 of file [scheduler.h](#).

Referenced by [l4_sched_cpu_set\(\)](#), and [set\(\)](#).

Here is the caller graph for this function:



14.208.3 Field Documentation

14.208.3.1 gran_offset

```
l4_umword_t l4_sched_cpu_set_t::gran_offset
```

Combination of granularity and offset.

The granularity defines how many CPUs each bit in map describes. And the offset is the number of the first CPU described by the first bit in the bitmap.

Precondition

offset must be a multiple of $2^{\text{granularity}}$.

MSB	LSB
8bit granularity	24bit offset ..

Definition at line 58 of file [scheduler.h](#).

Referenced by [L4::Scheduler::info\(\)](#), and [l4_sched_cpu_set\(\)](#).

The documentation for this struct was generated from the following file:

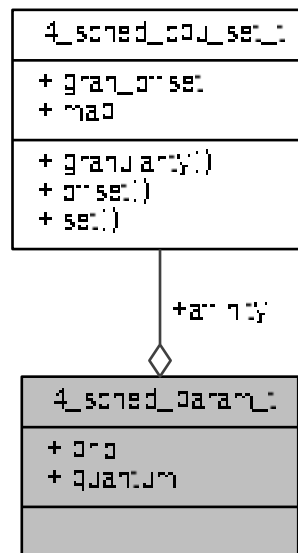
- [l4/sys/scheduler.h](#)

14.209 l4_sched_param_t Struct Reference

Scheduler parameter set.

```
#include <scheduler.h>
```

Collaboration diagram for l4_sched_param_t:



Data Fields

- [l4_sched_cpu_set_t affinity](#)
CPU affinity.
- [l4_umword_t prio](#)
Priority for scheduling.
- [l4_umword_t quantum](#)
Timeslice in micro seconds.

14.209.1 Detailed Description

Scheduler parameter set.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/migrate/thread_migrate.cc](#), [examples/sys/singlestep/main.c](#),
[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 120 of file [scheduler.h](#).

The documentation for this struct was generated from the following file:

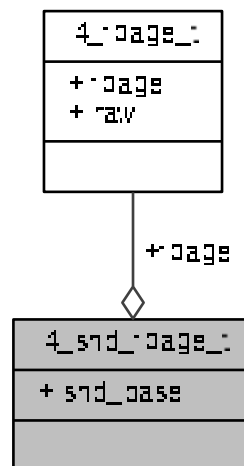
- [l4/sys/scheduler.h](#)

14.210 l4_snd_fpage_t Struct Reference

Send-flex-page types.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4_snd_fpage_t:



Data Fields

- [l4_umword_t snd_base](#)
Offset in receive window (send base)
- [l4_fpage_t fpage](#)
Source flex-page descriptor.

14.210.1 Detailed Description

Send-flex-page types.

Definition at line 98 of file [__l4_fpage.h](#).

The documentation for this struct was generated from the following file:

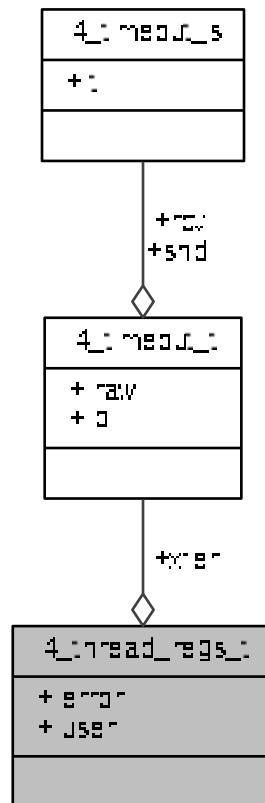
- `l4/sys/__l4_fpage.h`

14.211 l4_thread_regs_t Struct Reference

Encapsulation of the thread-control-register block of the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4_thread_regs_t:



Data Fields

- [l4_umword_t error](#)
System call error codes.
- [l4_timeout_t xfer](#)
Message transfer timeout.
- [l4_umword_t user](#) [3]
User values (ignored and preserved by the kernel)

14.211.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 110 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

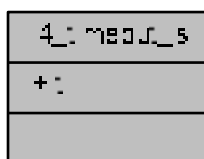
- [l4/sys/utcb.h](#)

14.212 l4_timeout_s Struct Reference

Basic timeout specification.

```
#include <__timeout.h>
```

Collaboration diagram for l4_timeout_s:



Data Fields

- [l4_uint16_t](#)
timeout value

14.212.1 Detailed Description

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m \cdot 2^e$).

The timeout can also specify an absolute point in time (bit 16 == 1).

Definition at line [45](#) of file [__timeout.h](#).

The documentation for this struct was generated from the following file:

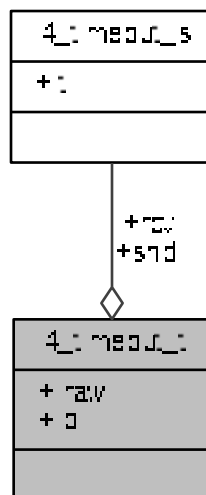
- [l4/sys/__timeout.h](#)

14.213 l4_timeout_t Union Reference

Timeout pair.

```
#include <__timeout.h>
```

Collaboration diagram for l4_timeout_t:



Data Fields

- [l4_uint32_t raw](#)
raw value
- ```

struct {
 l4_timeout_s rcv
 receive timeout
 l4_timeout_s snd
 send timeout
} p

```

*combined timeout*

### 14.213.1 Detailed Description

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

Definition at line 57 of file [\\_\\_timeout.h](#).

The documentation for this union was generated from the following file:

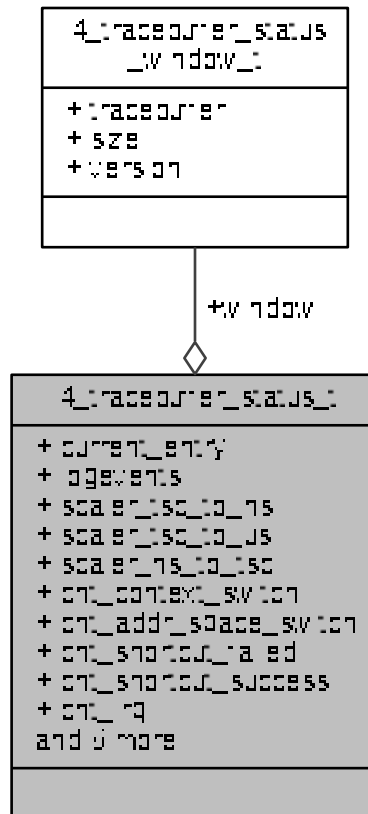
- [l4/sys/\\_\\_timeout.h](#)

## 14.214 l4\_tracebuffer\_status\_t Struct Reference

Trace-buffer status.

```
#include <ktrace.h>
```

Collaboration diagram for l4\_tracebuffer\_status\_t:



### Data Fields

- volatile l4\_tracebuffer\_entry\_t \* [current\\_entry](#)  
Address of the most current event in trace-buffer.
- l4\_uint32\_t [logevents](#) [LOG\_EVENT\_MAX\_EVENTS]  
Available LOG events.
- l4\_uint32\_t [scaler\\_tsc\\_to\\_ns](#)  
Scaler used for translation of CPU cycles to nano seconds.
- l4\_uint32\_t [scaler\\_tsc\\_to\\_us](#)  
Scaler used for translation of CPU cycles to micro seconds.
- l4\_uint32\_t [scaler\\_ns\\_to\\_tsc](#)  
Scaler used for translation of nano seconds to CPU cycles.

- volatile [l4\\_uint32\\_t cnt\\_context\\_switch](#)  
*Number of context switches (intra AS or inter AS)*
- volatile [l4\\_uint32\\_t cnt\\_addr\\_space\\_switch](#)  
*Number of inter AS context switches.*
- volatile [l4\\_uint32\\_t cnt\\_shortcut\\_failed](#)  
*How often was the IPC shortcut taken.*
- volatile [l4\\_uint32\\_t cnt\\_shortcut\\_success](#)  
*How often was the IPC shortcut not taken.*
- volatile [l4\\_uint32\\_t cnt\\_irq](#)  
*Number of hardware interrupts (without kernel scheduling interrupt)*
- volatile [l4\\_uint32\\_t cnt\\_ipc\\_long](#)  
*Number of long IPCs.*
- volatile [l4\\_uint32\\_t cnt\\_page\\_fault](#)  
*Number of page faults.*
- volatile [l4\\_uint32\\_t cnt\\_io\\_fault](#)  
*Number of faults (application runs at IOPL 0 and tries to execute cli, sti, in, or out but does not have a sufficient right in the I/O bitmap)*
- volatile [l4\\_uint32\\_t cnt\\_task\\_create](#)  
*Number of tasks created.*
- volatile [l4\\_uint32\\_t cnt\\_schedule](#)  
*Number of reschedules.*
- volatile [l4\\_uint32\\_t cnt\\_iobmap\\_tlb\\_flush](#)  
*Number of flushes of the I/O bitmap.*

### 14.214.1 Detailed Description

Trace-buffer status.

Definition at line 67 of file [ktrace.h](#).

### 14.214.2 Field Documentation

#### 14.214.2.1 cnt\_iobmap\_tlb\_flush

```
volatile l4_uint32_t l4_tracebuffer_status_t::cnt_iobmap_tlb_flush
```

Number of flushes of the I/O bitmap.

Increases on context switches between two small address spaces if at least one of the spaces has an I/O bitmap allocated.

Definition at line 106 of file [ktrace.h](#).

The documentation for this struct was generated from the following file:

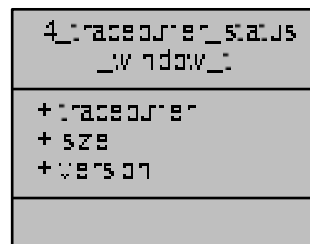
- [l4/sys/ktrace.h](#)

## 14.215 l4\_tracebuffer\_status\_window\_t Struct Reference

Trace-buffer status window descriptor.

```
#include <ktrace.h>
```

Collaboration diagram for l4\_tracebuffer\_status\_window\_t:



### Data Fields

- l4\_tracebuffer\_entry\_t \* [tracebuffer](#)  
*Address of trace-buffer.*
- [l4\\_umword\\_t](#) [size](#)  
*Size of trace-buffer.*
- volatile [l4\\_uint64\\_t](#) [version](#)  
*Version number of trace-buffer (incremented if trace-buffer overruns)*

### 14.215.1 Detailed Description

Trace-buffer status window descriptor.

Definition at line 52 of file [ktrace.h](#).

The documentation for this struct was generated from the following file:

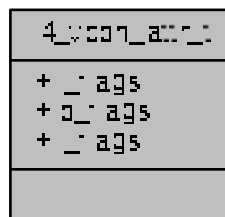
- l4/sys/[ktrace.h](#)

## 14.216 l4\_vcon\_attr\_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for l4\_vcon\_attr\_t:



### Data Fields

- [l4\\_umword\\_t i\\_flags](#)  
*input flags*
- [l4\\_umword\\_t o\\_flags](#)  
*output flags*
- [l4\\_umword\\_t l\\_flags](#)  
*local flags*

### 14.216.1 Detailed Description

Vcon attribute structure.

Definition at line 178 of file [vcon.h](#).

The documentation for this struct was generated from the following file:

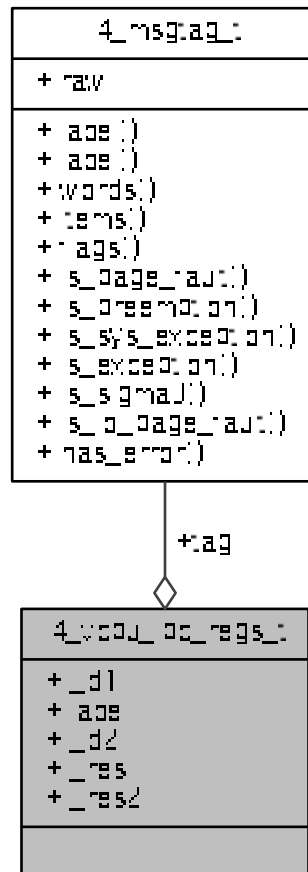
- [l4/sys/vcon.h](#)

## 14.217 l4\_vcpu\_ipc\_regs\_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_ipc\_regs\_t:



### 14.217.1 Detailed Description

vCPU message registers.

Definition at line 63 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following file:

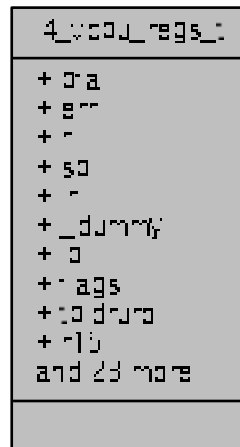
- arm/l4/sys/\_\_vcpu-arch.h

## 14.218 l4\_vcpu\_regs\_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_regs\_t:



### Data Fields

- [l4\\_umword\\_t pfa](#)  
*page fault address*
- [l4\\_umword\\_t err](#)  
*error code*
- [l4\\_umword\\_t sp](#)  
*stack pointer*
- [l4\\_umword\\_t ip](#)  
*instruction pointer*
- [l4\\_umword\\_t flags](#)  
*eflags*
- [l4\\_umword\\_t tpidru0](#)  
*Thread-ID register.*
- [l4\\_umword\\_t r15](#)  
*r15 register*
- [l4\\_umword\\_t r14](#)  
*r14 register*
- [l4\\_umword\\_t r13](#)  
*r13 register*
- [l4\\_umword\\_t r12](#)  
*r12 register*



- [l4\\_umword\\_t r11](#)  
*r11 register*
- [l4\\_umword\\_t r10](#)  
*r10 register*
- [l4\\_umword\\_t r9](#)  
*r9 register*
- [l4\\_umword\\_t r8](#)  
*r8 register*
- [l4\\_umword\\_t rdi](#)  
*rdi register*
- [l4\\_umword\\_t rsi](#)  
*rsi register*
- [l4\\_umword\\_t bp](#)  
*rbp register*
- [l4\\_umword\\_t bx](#)  
*rbx register*
- [l4\\_umword\\_t dx](#)  
*rdx register*
- [l4\\_umword\\_t cx](#)  
*rcx register*
- [l4\\_umword\\_t ax](#)  
*rax register*
- [l4\\_umword\\_t trapno](#)  
*trap number*
- [l4\\_umword\\_t cs](#)  
*dummy*
- [l4\\_umword\\_t ss](#)  
*ss register*
- [l4\\_umword\\_t es](#)  
*gs register*
- [l4\\_umword\\_t ds](#)  
*fs register*
- [l4\\_umword\\_t gs](#)  
*gs register*
- [l4\\_umword\\_t fs](#)  
*fs register*
- [l4\\_umword\\_t dummy1](#)  
*dummy*

### 14.218.1 Detailed Description

vCPU registers.

Definition at line 39 of file [\\_\\_vcpu-arch.h](#).

### 14.218.2 Field Documentation

**14.218.2.1 ax**

`l4_umword_t l4_vcpu_regs_t::ax`

rax register

eax register

Definition at line 68 of file [\\_\\_vcpu-arch.h](#).

**14.218.2.2 bp**

`l4_umword_t l4_vcpu_regs_t::bp`

rbp register

ebp register

Definition at line 63 of file [\\_\\_vcpu-arch.h](#).

**14.218.2.3 bx**

`l4_umword_t l4_vcpu_regs_t::bx`

rbx register

ebx register

Definition at line 65 of file [\\_\\_vcpu-arch.h](#).

**14.218.2.4 cx**

`l4_umword_t l4_vcpu_regs_t::cx`

rcx register

ecx register

Definition at line 67 of file [\\_\\_vcpu-arch.h](#).

#### 14.218.2.5 di

`l4_umword_t l4_vcpu_regs_t::di`

rdi register

edi register

Definition at line 61 of file [\\_\\_vcpu-arch.h](#).

#### 14.218.2.6 dx

`l4_umword_t l4_vcpu_regs_t::dx`

rdx register

edx register

Definition at line 66 of file [\\_\\_vcpu-arch.h](#).

#### 14.218.2.7 si

`l4_umword_t l4_vcpu_regs_t::si`

rsi register

esi register

Definition at line 62 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following file:

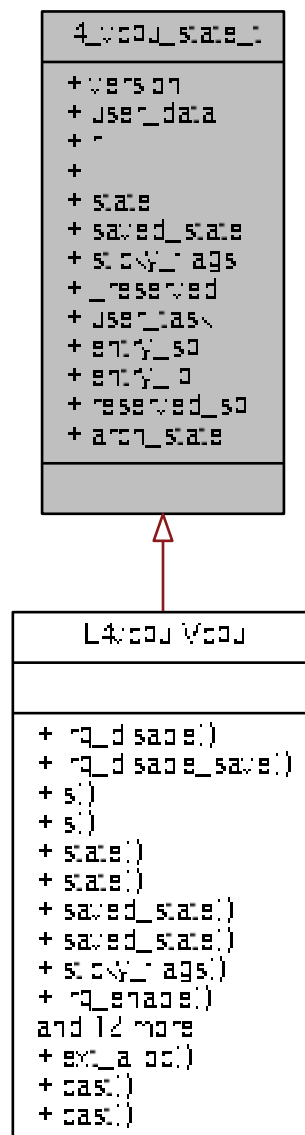
- [arm/l4/sys/\\_\\_vcpu-arch.h](#)

## 14.219 l4\_vcpu\_state\_t Struct Reference

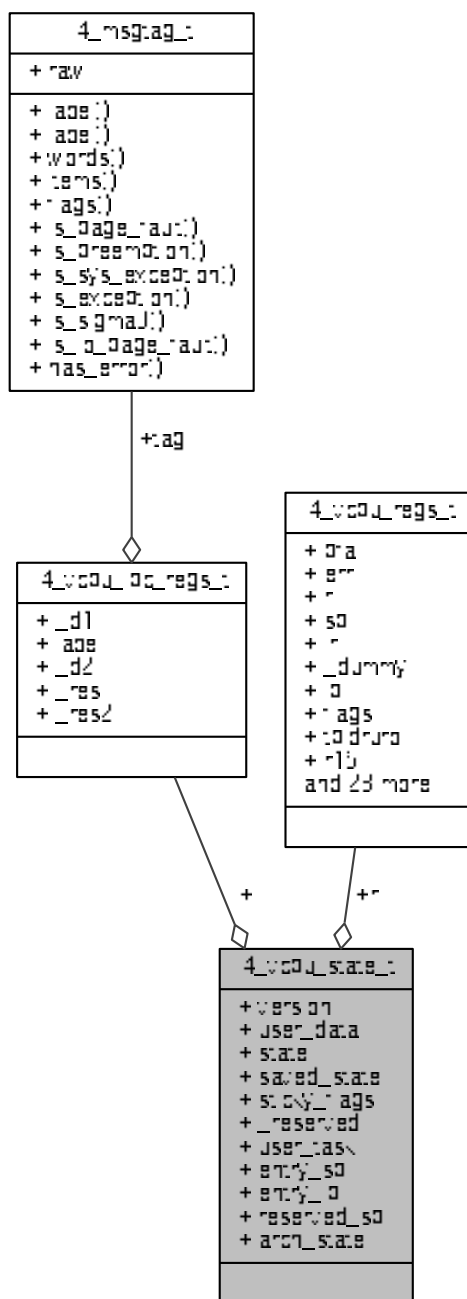
State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for l4\_vcpu\_state\_t:



Collaboration diagram for l4\_vcpu\_state\_t:



## Data Fields

- [l4\\_vcpu\\_regs\\_t r](#)  
*Register state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t i](#)  
*IPC state.*
- [l4\\_uint16\\_t state](#)

*Current vCPU state.*

- [l4\\_uint16\\_t saved\\_state](#)

*Saved vCPU state.*

- [l4\\_uint16\\_t sticky\\_flags](#)

*Pending flags.*

- [l4\\_cap\\_idx\\_t user\\_task](#)

*User task to use.*

- [l4\\_umword\\_t entry\\_sp](#)

*Stack pointer for entry (when coming from user task)*

- [l4\\_umword\\_t entry\\_ip](#)

*IP for entry.*

### 14.219.1 Detailed Description

State of a vCPU.

Definition at line 47 of file [vcpu.h](#).

The documentation for this struct was generated from the following file:

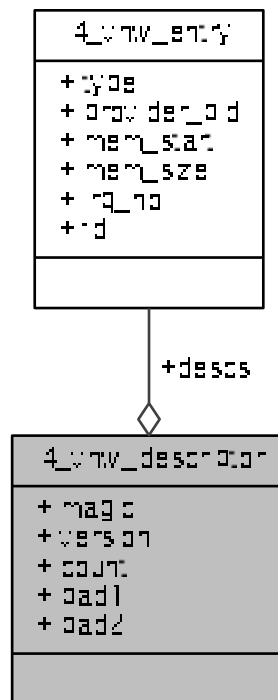
- [l4/sys/vcpu.h](#)

## 14.220 l4\_vhw\_descriptor Struct Reference

Virtual hardware devices description.

```
#include <vhw.h>
```

Collaboration diagram for l4\_vhw\_descriptor:



## Data Fields

- [l4\\_uint32\\_t magic](#)  
*Magic.*
- [l4\\_uint8\\_t version](#)  
*Version of the descriptor.*
- [l4\\_uint8\\_t count](#)  
*Number of entries.*
- [l4\\_uint8\\_t pad1](#)  
*padding*
- [l4\\_uint8\\_t pad2](#)  
*padding*
- `struct l4_vhw_entry descs []`  
*Array of device descriptions.*

### 14.220.1 Detailed Description

Virtual hardware devices description.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 70 of file [vhw.h](#).

## 14.220.2 Field Documentation

### 14.220.2.1 count

```
l4_uint8_t l4_vhw_descriptor::count
```

Number of entries.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 73 of file [vhw.h](#).

### 14.220.2.2 descs

```
struct l4_vhw_entry l4_vhw_descriptor::descs[]
```

Array of device descriptions.

Definition at line 77 of file [vhw.h](#).

### 14.220.2.3 magic

```
l4_uint32_t l4_vhw_descriptor::magic
```

Magic.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 71 of file [vhw.h](#).



## 14.220.2.4 version

```
l4_uint8_t l4_vhw_descriptor::version
```

Version of the descriptor.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 72 of file [vhw.h](#).

The documentation for this struct was generated from the following file:

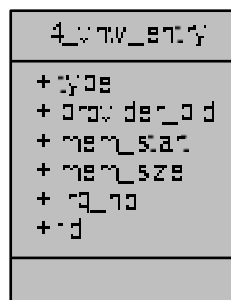
- [l4/sys/vhw.h](#)

## 14.221 l4\_vhw\_entry Struct Reference

Description of a device.

```
#include <vhw.h>
```

Collaboration diagram for l4\_vhw\_entry:



### Data Fields

- enum [l4\\_vhw\\_entry\\_type](#) `type`  
*Type of virtual hardware.*
- [l4\\_uint32\\_t](#) `provider_pid`  
*Host PID of the VHW provider.*
- [l4\\_addr\\_t](#) `mem_start`  
*Start of memory region.*
- [l4\\_addr\\_t](#) `mem_size`  
*Size of memory region.*
- [l4\\_uint32\\_t](#) `irq_no`  
*IRQ number.*
- [l4\\_uint32\\_t](#) `fd`  
*File descriptor.*

### 14.221.1 Detailed Description

Description of a device.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 55 of file [vhw.h](#).

### 14.221.2 Field Documentation

#### 14.221.2.1 fd

```
l4_uint32_t l4_vhw_entry::fd
```

File descriptor.

Definition at line 63 of file [vhw.h](#).

#### 14.221.2.2 irq\_no

```
l4_uint32_t l4_vhw_entry::irq_no
```

IRQ number.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 62 of file [vhw.h](#).

#### 14.221.2.3 mem\_size

```
l4_addr_t l4_vhw_entry::mem_size
```

Size of memory region.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 60 of file [vhw.h](#).

#### 14.221.2.4 mem\_start

```
l4_addr_t l4_vhw_entry::mem_start
```

Start of memory region.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 59 of file [vhw.h](#).

#### 14.221.2.5 provider\_pid

```
l4_uint32_t l4_vhw_entry::provider_pid
```

Host PID of the VHW provider.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 57 of file [vhw.h](#).

#### 14.221.2.6 type

```
enum l4_vhw_entry_type l4_vhw_entry::type
```

Type of virtual hardware.

Examples:

[examples/sys/ux-vhw/main.c](#).

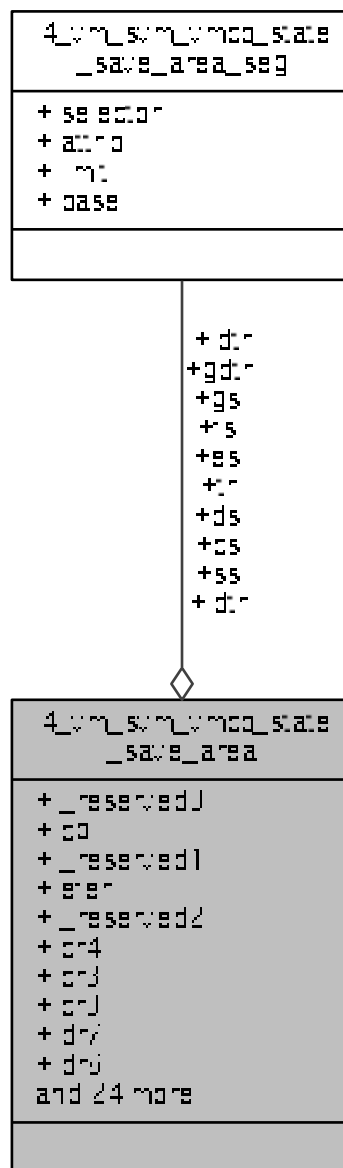
Definition at line 56 of file [vhw.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/vhw.h](#)



Collaboration diagram for l4\_vm\_svm\_vmcb\_state\_save\_area:



### 14.223.1 Detailed Description

State save area structure for SVM VMs.

Definition at line 96 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

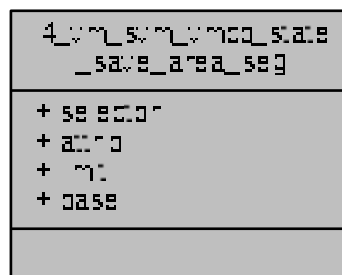
- l4/sys/\_\_vm-svm.h

## 14.224 l4\_vm\_svm\_vmcb\_state\_save\_area\_seg Struct Reference

State save area segment selector struct.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_state\_save\_area\_seg:



### 14.224.1 Detailed Description

State save area segment selector struct.

Definition at line 84 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

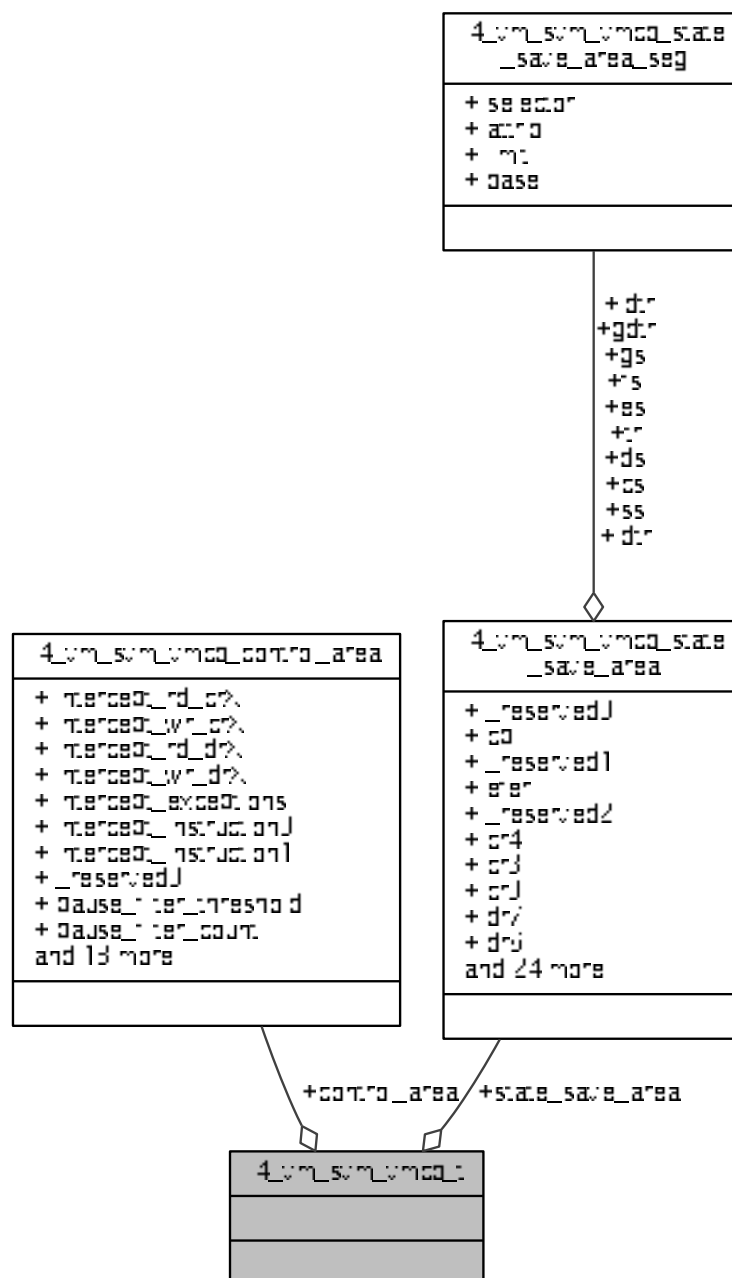
- l4/sys/\_\_vm-svm.h

## 14.225 l4\_vm\_svm\_vmcb\_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_t:



### 14.225.1 Detailed Description

Control structure for SVM VMs.

Definition at line 165 of file `__vm-svm.h`.

The documentation for this struct was generated from the following file:

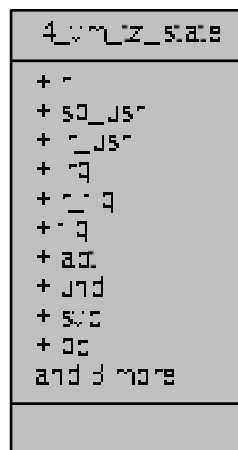
- `l4/sys/__vm-svm.h`

## 14.226 l4\_vm\_tz\_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4\_vm\_tz\_state:



### 14.226.1 Detailed Description

state structure for TrustZone VMs

Definition at line 52 of file [vm.h](#).

The documentation for this struct was generated from the following file:

- [arm/l4/sys/vm.h](#)

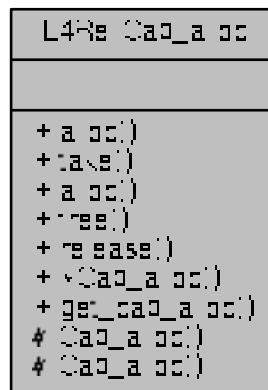
## 14.227 L4Re::Cap\_alloc Class Reference

Capability allocator interface.

Inherited by `L4Re::Cap_alloc_t< ALLOC >`.



Collaboration diagram for L4Re::Cap\_alloc:



## Public Member Functions

- virtual [L4::Cap](#)< void > [alloc](#) ()=0 throw ()  
*Allocate a capability.*
- template<typename T >  
[L4::Cap](#)< T > [alloc](#) () throw ()  
*Allocate a capability.*
- virtual void [free](#) ([L4::Cap](#)< void > cap, [l4\\_cap\\_idx\\_t](#) task=[L4\\_INVALID\\_CAP](#), unsigned unmap\_flags=[L4\\_↔  
FP\\_ALL\\_SPACES](#))=0 throw ()  
*Free a capability.*
- virtual [~Cap\\_alloc](#) ()=0  
*Destructor.*

## Static Public Member Functions

- template<typename CAP\_ALLOC >  
static [L4Re::Cap\\_alloc](#) \* [get\\_cap\\_alloc](#) (CAP\_ALLOC &ca)  
*Construct an instance of a capability allocator.*

### 14.227.1 Detailed Description

Capability allocator interface.

Definition at line 41 of file [cap\\_alloc](#).

### 14.227.2 Member Function Documentation

14.227.2.1 `alloc()` [1/2]

```
virtual L4::Cap<void> L4Re::Cap_alloc::alloc () throw () [pure virtual]
```

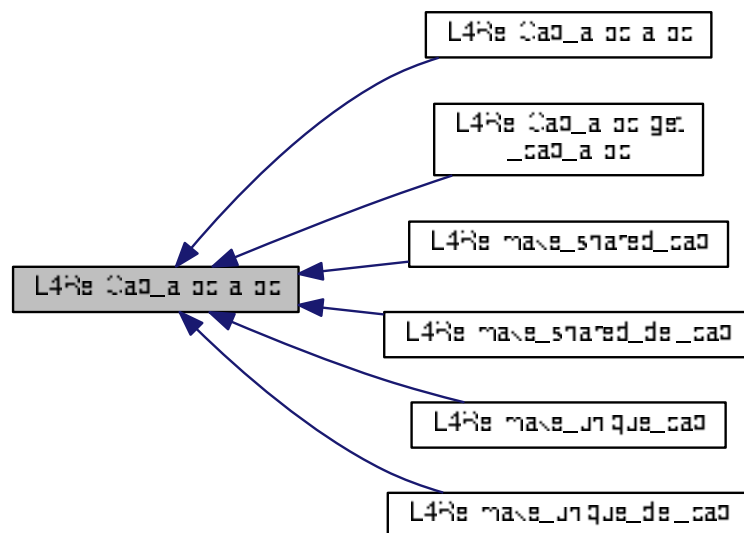
Allocate a capability.

## Returns

Capability of type void

Referenced by [alloc\(\)](#), [get\\_cap\\_alloc\(\)](#), [L4Re::make\\_shared\\_cap\(\)](#), [L4Re::make\\_shared\\_del\\_cap\(\)](#), [L4Re::make\\_shared\\_unique\\_cap\(\)](#), and [L4Re::make\\_unique\\_del\\_cap\(\)](#).

Here is the caller graph for this function:

14.227.2.2 `alloc()` [2/2]

```
template<typename T>
L4::Cap<T> L4Re::Cap_alloc::alloc () throw () [inline]
```

Allocate a capability.

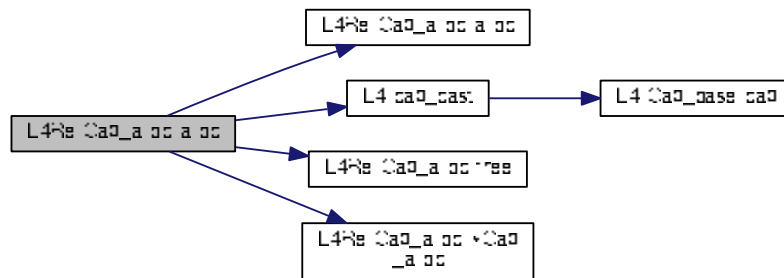
## Returns

Capability of type T

Definition at line 64 of file [cap\\_alloc](#).

References [alloc\(\)](#), [L4::cap\\_cast\(\)](#), [free\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), [L4\\_INVALID\\_CAP](#), and [~Cap\\_alloc\(\)](#).

Here is the call graph for this function:



## 14.227.2.3 free()

```

virtual void L4Re::Cap_alloc::free (
 L4::Cap< void > cap,
 l4_cap_idx_t task = L4_INVALID_CAP,
 unsigned unmap_flags = L4_FP_ALL_SPACES) throw () [pure virtual]

```

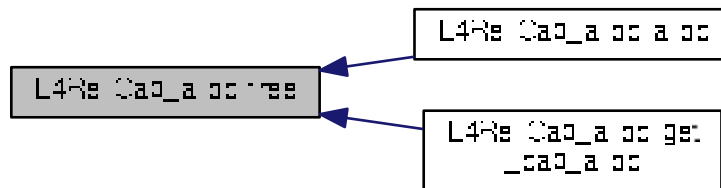
Free a capability.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <i>cap</i>         | Capability to free.                                     |
| <i>task</i>        | If set, task to unmap the capability from.              |
| <i>unmap_flags</i> | Flags for unmap, see <a href="#">l4_unmap_flags_t</a> . |

Referenced by [alloc\(\)](#), and [get\\_cap\\_alloc\(\)](#).

Here is the caller graph for this function:



#### 14.227.2.4 get\_cap\_alloc()

```
template<typename CAP_ALLOC >
static L4Re::Cap_alloc* L4Re::Cap_alloc::get_cap_alloc (
 CAP_ALLOC & ca) [inline], [static]
```

Construct an instance of a capability allocator.

##### Parameters

|           |                      |
|-----------|----------------------|
| <i>ca</i> | Capability allocator |
|-----------|----------------------|

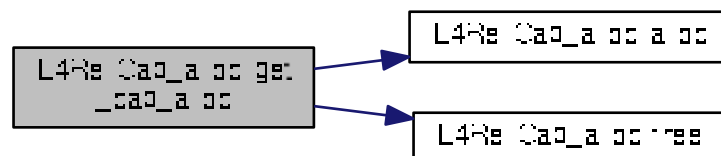
##### Returns

Instance of a capability allocator.

Definition at line 90 of file [cap\\_alloc](#).

References [alloc\(\)](#), [free\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), and [L4\\_INVALID\\_CAP](#).

Here is the call graph for this function:



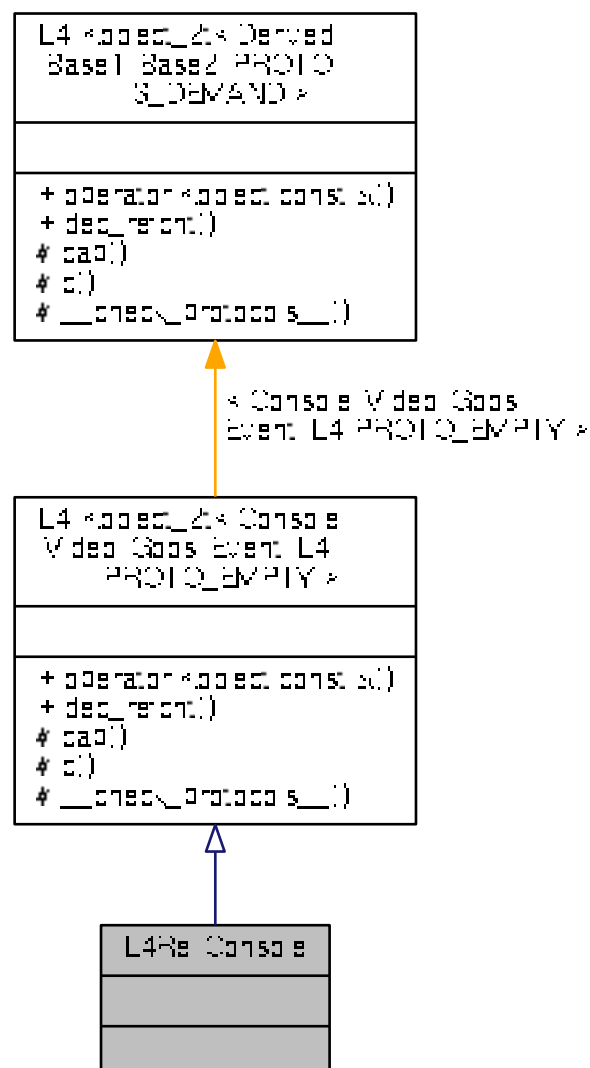
The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

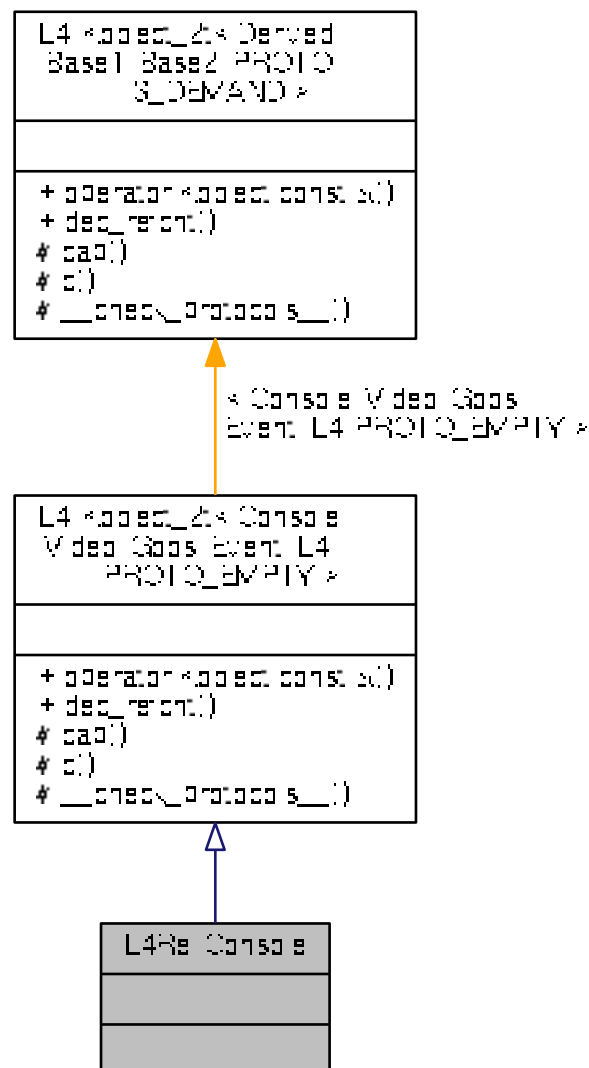
## 14.228 L4Re::Console Class Reference

[Console](#) class.

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



## Additional Inherited Members

### 14.228.1 Detailed Description

[Console](#) class.

Definition at line 39 of file [console](#).

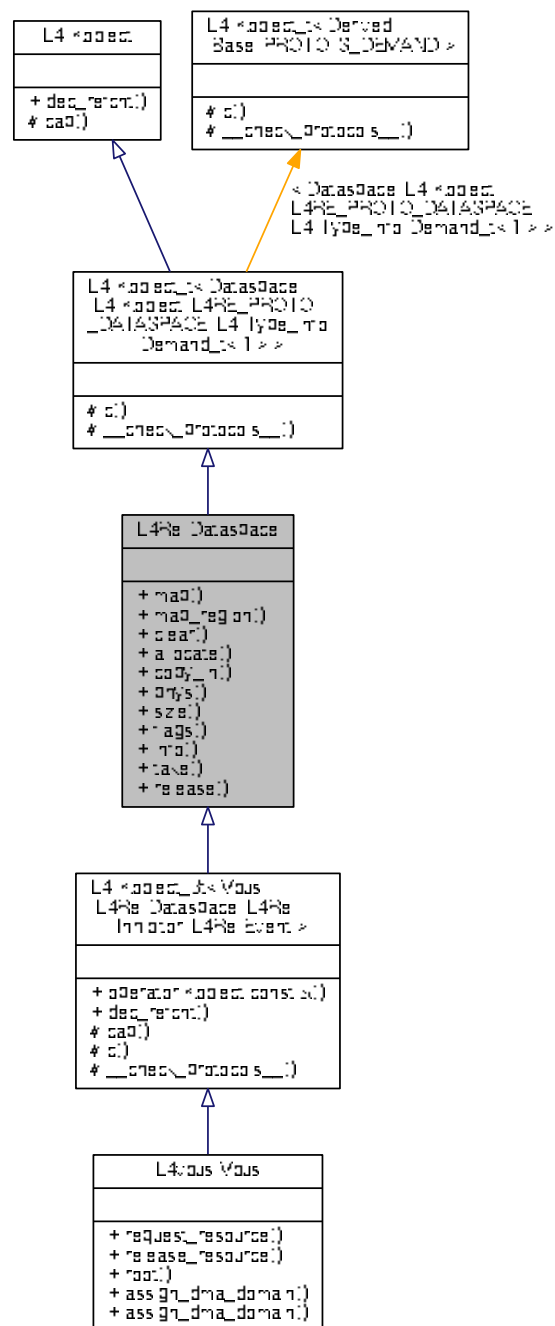
The documentation for this class was generated from the following file:

- [l4/re/console](#)

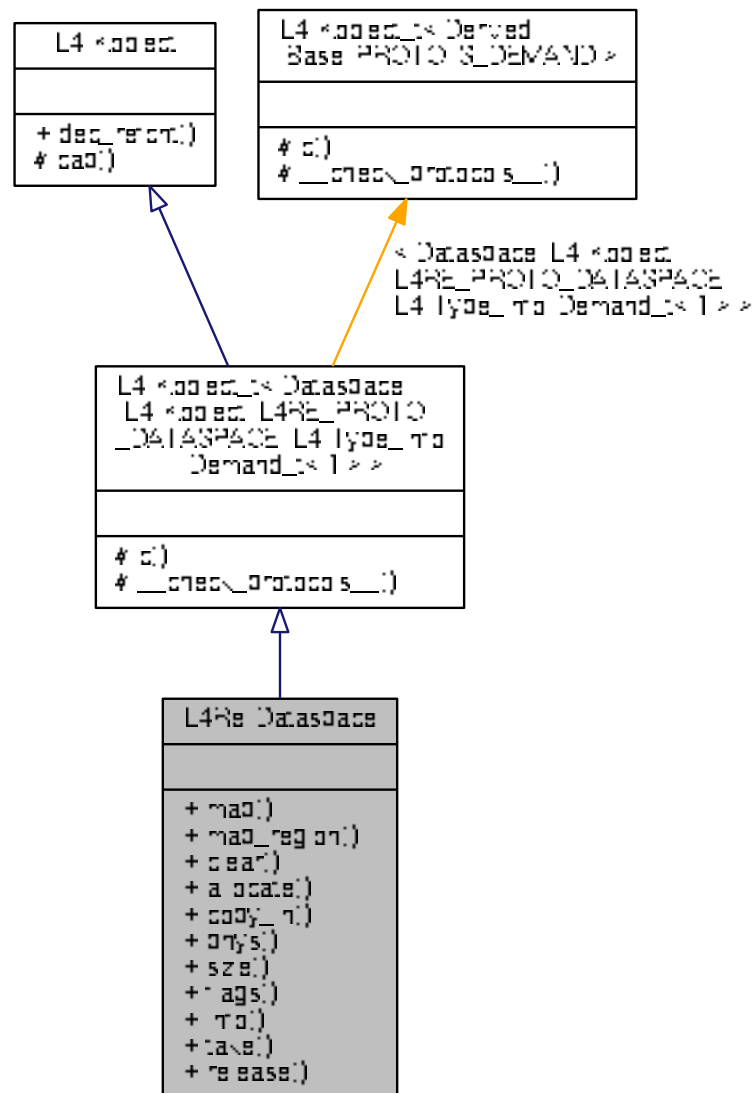
## 14.229 L4Re::Dataspace Class Reference

Interface for memory-like objects.

Inheritance diagram for L4Re::Dataspace:



Collaboration diagram for L4Re::Dataspace:



## Data Structures

- struct [Stats](#)

*Information about the dataspace.*

## Public Types

- enum [Map\\_flags](#) {  
`Map_ro = 0, Map_rw = 1, Map_normal = 0x00, Map_cacheable = Map_normal,`  
`Map_bufferable = 0x10, Map_uncacheable = 0x20, Map_caching_mask = 0x30, Map_caching_shift = 4 }`

*Flags for map operations.*



## Public Member Functions

- long [map](#) ([l4\\_addr\\_t](#) offset, unsigned long [flags](#), [l4\\_addr\\_t](#) local\_addr, [l4\\_addr\\_t](#) min\_addr, [l4\\_addr\\_t](#) max\_↵  
addr) const throw ()  
*Request a flex-page mapping from the dataspace.*
- long [map\\_region](#) ([l4\\_addr\\_t](#) offset, unsigned long [flags](#), [l4\\_addr\\_t](#) min\_addr, [l4\\_addr\\_t](#) max\_addr) const throw  
( )  
*Map a part of a dataspace completely.*
- long [clear](#) ([l4\\_addr\\_t](#) offset, unsigned long [size](#))  
*Clear parts of a dataspace.*
- long [allocate](#) ([l4\\_addr\\_t](#) offset, [l4\\_size\\_t](#) size)  
*Allocate a range in the dataspace.*
- long [copy\\_in](#) ([l4\\_addr\\_t](#) dst\_offs, [L4::lpc::Cap](#)< [Dataspace](#) > src, [l4\\_addr\\_t](#) src\_offs, unsigned long [size](#))  
*Copy contents from another dataspace.*
- long [phys](#) ([l4\\_addr\\_t](#) offset, [l4\\_addr\\_t](#) &phys\_addr, [l4\\_size\\_t](#) &phys\_size)  
*Get the physical addresses of a dataspace.*
- unsigned long [size](#) () const throw ()  
*Get size of a dataspace.*
- long [flags](#) () const throw ()  
*Get flags of the dataspace.*
- long [info](#) ([Stats](#) \*stats)  
*Get information on the dataspace.*

## Additional Inherited Members

### 14.229.1 Detailed Description

Interface for memory-like objects.

Dataspaces are a central abstraction provided by [L4Re](#). A dataspace is an abstraction for any thing that is available via usual memory access instructions. A dataspace can be a file, as well as the memory-mapped registers of a device, or anonymous memory, such as a heap.

The dataspace interface defines a set of methods that allow any kind of dataspace to be attached (mapped) to the virtual address space of an [L4](#) task and then be accessed via memory-access instructions. The [L4Re::Rm](#) interface can be used to attach a dataspace to a virtual address space of a task paged by a certain instance of a region map.

#### Include File

```
#include <l4/re/dataspace>
```

#### Examples:

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared+\\_ds/ds\\_srv.cc](#).

Definition at line 59 of file [dataspace](#).

### 14.229.2 Member Enumeration Documentation

#### 14.229.2.1 Map\_flags

```
enum L4Re::Dataspace::Map_flags
```

Flags for map operations.

## Enumerator

|                   |                                              |
|-------------------|----------------------------------------------|
| Map_ro            | Request read-only mapping.                   |
| Map_rw            | Request writable mapping.                    |
| Map_normal        | request normal memory mapping                |
| Map_cacheable     | request normal memory mapping                |
| Map_bufferable    | request bufferable (write buffered) mappings |
| Map_uncacheable   | request uncacheable memory mappings          |
| Map_caching_mask  | mask for caching flags                       |
| Map_caching_shift | shift value for caching flags                |

Definition at line 68 of file [dataspace](#).

## 14.229.3 Member Function Documentation

## 14.229.3.1 allocate()

```
long L4Re::Dataspace::allocate (
 l4_addr_t offset,
 l4_size_t size)
```

Allocate a range in the dataspace.

## Parameters

|               |                                    |
|---------------|------------------------------------|
| <i>offset</i> | Offset in the dataspace, in bytes. |
| <i>size</i>   | Size of the range, in bytes.       |

## Return values

|                   |                                                                                                                    |
|-------------------|--------------------------------------------------------------------------------------------------------------------|
| <i>L4_EOK</i>     | Success                                                                                                            |
| <i>-L4_ERANGE</i> | Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.) |
| <i>-L4_ENOMEM</i> | Not enough memory available.                                                                                       |
| <i>&lt;0</i>      | IPC errors                                                                                                         |

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

## 14.229.3.2 clear()

```
long L4Re::Dataspace::clear (
 l4_addr_t offset,
 unsigned long size)
```

Clear parts of a dataspace.

#### Parameters

|               |                                     |
|---------------|-------------------------------------|
| <i>offset</i> | Offset within dataspace (in bytes). |
| <i>size</i>   | Size of region to clear (in bytes). |

#### Return values

|                          |                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------|
| $\geq 0$                 | Success.                                                                                                           |
| <code>-L4_ERANGE</code>  | Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.) |
| <code>-L4_EACCESS</code> | <a href="#">Dataspace</a> is read-only.                                                                            |
| $< 0$                    | IPC errors                                                                                                         |

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

#### 14.229.3.3 `copy_in()`

```
long L4Re::Dataspace::copy_in (
 l4_addr_t dst_offs,
 L4::Ipc::Cap< Dataspace > src,
 l4_addr_t src_offs,
 unsigned long size)
```

Copy contents from another dataspace.

#### Parameters

|                 |                                  |
|-----------------|----------------------------------|
| <i>dst_offs</i> | Offset in destination dataspace. |
| <i>src</i>      | Source dataspace to copy from.   |
| <i>src_offs</i> | Offset in the source dataspace.  |
| <i>size</i>     | Size to copy (in bytes).         |

#### Return values

|                          |                                     |
|--------------------------|-------------------------------------|
| <code>L4_EOK</code>      | Success                             |
| <code>-L4_EACCESS</code> | Destination dataspace not writable. |
| <code>-L4_EINVAL</code>  | Invalid parameter supplied.         |
| $< 0$                    | IPC errors                          |

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspace managers are not from the same dataspace manager or the dataspace managers do not cooperate.

#### 14.229.3.4 `flags()`

```
long L4Re::Dataspace::flags () const throw)
```

Get flags of the dataspace.

## Return values

|          |                        |
|----------|------------------------|
| $\geq 0$ | Flags of the dataspace |
| $< 0$    | IPC errors             |

## See also

[L4Re::Dataspace::Map\\_flags](#)

Definition at line 116 of file [dataspace\\_impl.h](#).

## 14.229.3.5 info()

```
long L4Re::Dataspace::info (
 Stats * stats)
```

Get information on the dataspace.

## Parameters

|     |       |                                       |
|-----|-------|---------------------------------------|
| out | stats | <a href="#">Dataspace</a> information |
|-----|-------|---------------------------------------|

## Return values

|       |            |
|-------|------------|
| 0     | Success    |
| $< 0$ | IPC errors |

## 14.229.3.6 map()

```
long L4Re::Dataspace::map (
 l4_addr_t offset,
 unsigned long flags,
 l4_addr_t local_addr,
 l4_addr_t min_addr,
 l4_addr_t max_addr) const throw ()
```

Request a flex-page mapping from the dataspace.

## Parameters

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <i>offset</i>     | Offset to start within dataspace                              |
| <i>flags</i>      | map flags, see <a href="#">Map_flags</a> .                    |
| <i>local_addr</i> | Local address to map to.                                      |
| <i>min_addr</i>   | Defines start of receive window. (Rounded down to page size.) |
| <i>max_addr</i>   | Defines end of receive window. (Rounded up to page size.)     |

## Return values

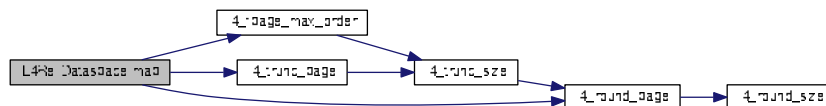
|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>L4_EOK</i>     | Success                                               |
| <i>-L4_ERANGE</i> | Invalid offset.                                       |
| <i>-L4_EPERM</i>  | Insufficient permission to map with requested rights. |
| <i>&lt;0</i>      | IPC errors                                            |

The map call will attempt to map the largest possible flexpage that covers the given local address and still fits into the region defined by `min_addr` and `max_addr`. If the given region is invalid or does not overlap the local address, the smallest valid page size is used.

Definition at line 92 of file [dataspace\\_impl.h](#).

References [l4\\_fpage\\_max\\_order\(\)](#), [L4\\_LOG2\\_PAGESIZE](#), [l4\\_round\\_page\(\)](#), and [l4\\_trunc\\_page\(\)](#).

Here is the call graph for this function:



## 14.229.3.7 map\_region()

```

long L4Re::Dataspace::map_region (
 l4_addr_t offset,
 unsigned long flags,
 l4_addr_t min_addr,
 l4_addr_t max_addr) const throw ()

```

Map a part of a dataspace completely.

## Parameters

|                 |                                                               |
|-----------------|---------------------------------------------------------------|
| <i>offset</i>   | Offset to start within dataspace                              |
| <i>flags</i>    | map flags, see <a href="#">Map_flags</a> .                    |
| <i>min_addr</i> | Defines start of receive window. (Rounded down to page size.) |
| <i>max_addr</i> | Defines end of receive window. (Rounded up to page size.)     |

## Return values

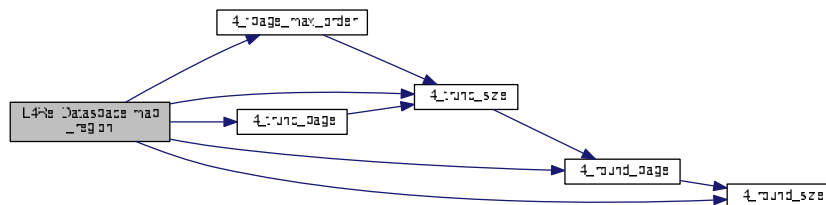
|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>L4_EOK</i>     | Success                                               |
| <i>-L4_ERANGE</i> | Invalid offset.                                       |
| <i>-L4_EPERM</i>  | Insufficient permission to map with requested rights. |
| <i>&lt;0</i>      | IPC errors                                            |

Definition at line 55 of file [dataspace\\_impl.h](#).

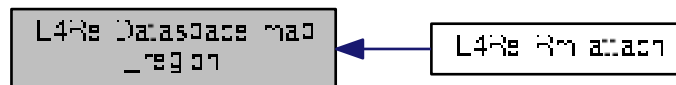
References [l4\\_fpage\\_max\\_order\(\)](#), [L4\\_LOG2\\_PAGESIZE](#), [l4\\_round\\_page\(\)](#), [l4\\_round\\_size\(\)](#), [l4\\_trunc\\_page\(\)](#), [l4\\_trunc\\_size\(\)](#), and [L4\\_UNLIKELY](#).

Referenced by [L4Re::Rm::attach\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.229.3.8 phys()

```

long L4Re::Dataspace::phys (
 l4_addr_t offset,
 l4_addr_t & phys_addr,
 l4_size_t & phys_size)

```

Get the physical addresses of a dataspace.

#### Parameters

|               |                     |
|---------------|---------------------|
| <i>offset</i> | Offset in dataspace |
|---------------|---------------------|

#### Return values

|                  |                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------|
| <i>phys_addr</i> | Physical address.                                                                                |
| <i>phys_size</i> | Size of largest physically contiguous region in the dataspace after <i>phys_addr</i> (in bytes). |

## Return values

|                         |                                          |
|-------------------------|------------------------------------------|
| <code>L4_EOK</code>     | Success                                  |
| <code>-L4_EINVAL</code> | <a href="#">Dataspace</a> is not pinned. |
| <code>&lt;0</code>      | IPC errors                               |

This call will only succeed on pinned memory dataspace.

**Deprecated** Use [L4Re::Dma\\_space](#) instead. This function will be removed in future releases.

14.229.3.9 `size()`

```
unsigned long L4Re::Dataspace::size () const throw ()
```

Get size of a dataspace.

## Returns

Size of the dataspace in bytes.

## Examples:

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line [106](#) of file [dataspace\\_impl.h](#).

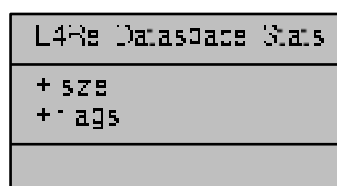
The documentation for this class was generated from the following files:

- [l4re/dataspace](#)
- [l4re/impl/dataspace\\_impl.h](#)

14.230 **L4Re::Dataspace::Stats Struct Reference**

Information about the dataspace.

Collaboration diagram for L4Re::Dataspace::Stats:





## Data Fields

- unsigned long [size](#)
- unsigned long [flags](#)

### 14.230.1 Detailed Description

Information about the dataspace.

Definition at line 85 of file [dataspace](#).

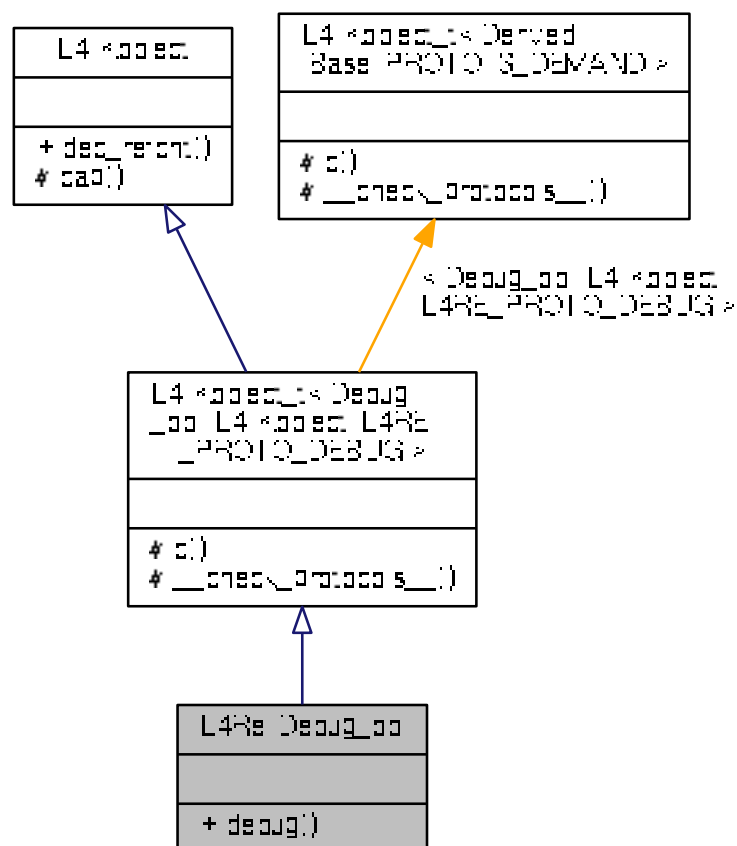
The documentation for this struct was generated from the following file:

- [l4/re/dataspace](#)

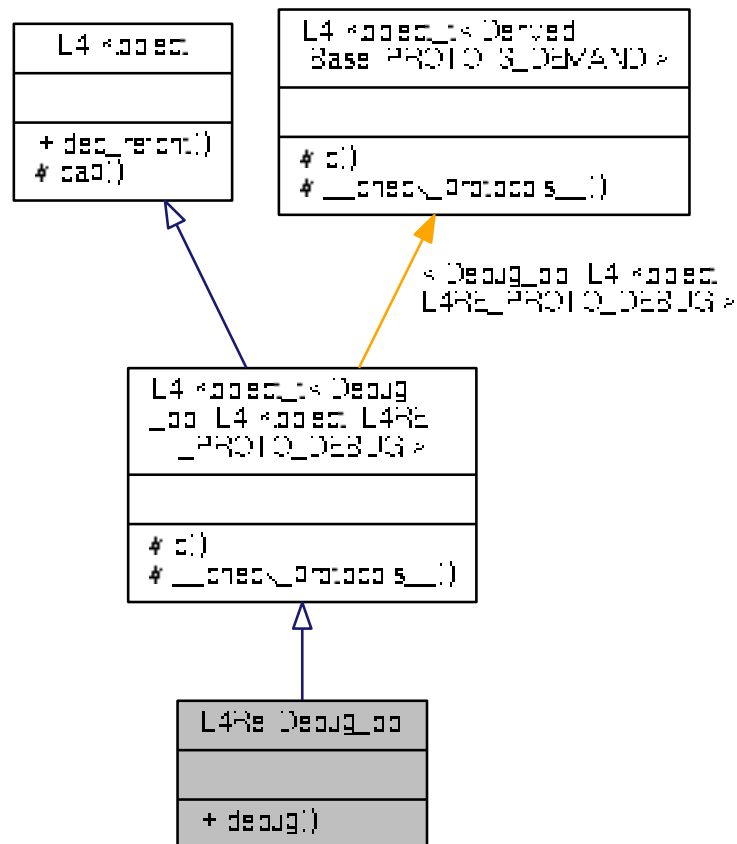
## 14.231 L4Re::Debug\_obj Class Reference

Debug interface.

Inheritance diagram for L4Re::Debug\_obj:



Collaboration diagram for L4Re::Debug\_obj:



## Public Member Functions

- long [debug](#) (unsigned long function)  
*Debug call.*

## Additional Inherited Members

### 14.231.1 Detailed Description

Debug interface.

See also

[Debugging API](#).

Definition at line 51 of file [debug](#).

## 14.231.2 Member Function Documentation

### 14.231.2.1 debug()

```
long L4Re::Debug_obj::debug (
 unsigned long function)
```

Debug call.

#### Parameters

|                 |                   |
|-----------------|-------------------|
| <i>function</i> | Function to call. |
|-----------------|-------------------|

#### Returns

- L4\_EOK
- IPC errors

The documentation for this class was generated from the following file:

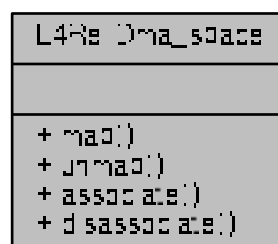
- [l4/re/debug](#)

## 14.232 L4Re::Dma\_space Class Reference

DMA Address Space.

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

Collaboration diagram for L4Re::Dma\_space:



## Public Types

- enum [Direction](#) { [Bidirectional](#), [To\\_device](#), [From\\_device](#), [None](#) }  
*Direction of the DMA transfers.*
- enum [Attribute](#) { [No\\_sync](#) }  
*Attributes used for the memory region during the transfer.*
- enum [Space\\_attr](#) { [Coherent](#), [Phys\\_space](#) }  
*Attributes assigned to the DMA space when associated with a specific device.*
- typedef [l4\\_uint64\\_t](#) [Dma\\_addr](#)  
*Data type for DMA addresses.*
- typedef [L4::Types::Flags](#)< [Attribute](#) > [Attributes](#)  
*Attributes for DMA mappings.*
- typedef [L4::Types::Flags](#)< [Space\\_attr](#) > [Space\\_attr](#)  
*Attributes used when configuring the DMA space.*

## Public Member Functions

- long [map](#) ([L4::lpc::Cap](#)< [L4Re::Dataspace](#) > src, [l4\\_addr\\_t](#) offset, [L4::lpc::In\\_out](#)< [l4\\_size\\_t](#) \*> size, [Attributes](#) attrs, [Direction](#) dir, [Dma\\_addr](#) \*dma\_addr)  
*Map the given part of this data space into the DMA address space.*
- long [unmap](#) ([Dma\\_addr](#) dma\_addr, [l4\\_size\\_t](#) size, [Attributes](#) attrs, [Direction](#) dir)  
*Unmap the given part of this data space from the DMA address space.*
- long [associate](#) ([L4::lpc::Opt](#)< [L4::lpc::Cap](#)< [L4::Task](#) > > dma\_task, [Space\\_attr](#) attr)  
*Associate a DMA task for a device to this [Dma\\_space](#).*
- long [disassociate](#) ()  
*Disassociate the DMA task from this [Dma\\_space](#).*

### 14.232.1 Detailed Description

DMA Address Space.

A [Dma\\_space](#) represents the DMA address space of a device. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a DMA address space, using [map\(\)](#), makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, [map\(\)](#) is responsible for the necessary syncing operations before the DMA.

[unmap\(\)](#) is the reverse operation to [map\(\)](#) and unmaps the given data-space part for the DMA address space. [unmap\(\)](#) is responsible for the necessary sync operations after the DMA.

Definition at line 61 of file [dma\\_space](#).

### 14.232.2 Member Typedef Documentation

### 14.232.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

Attributes for DMA mappings.

See also

[Attribute](#)

Definition at line 106 of file [dma\\_space](#).

### 14.232.3 Member Enumeration Documentation

#### 14.232.3.1 Attribute

```
enum L4Re::Dma_space::Attribute
```

Attributes used for the memory region during the transfer.

See also

[Attributes](#)

#### Enumerator

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| No_sync | Do not sync the memory hierarchy. When this flag is <i>not set</i> (default) the memory region shall be made coherent to the point-of-coherency of the device associated with this <a href="#">Dma_space</a> . When using this attribute the client is responsible for syncing the memory hierarchy for DMA. This can either be done using the cache API or by another <a href="#">map()</a> or <a href="#">unmap()</a> operation of the same part of the data space (without the <a href="#">No_sync</a> attribute). |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 85 of file [dma\\_space](#).

#### 14.232.3.2 Direction

```
enum L4Re::Dma_space::Direction
```

Direction of the DMA transfers.

#### Enumerator

|               |                                              |
|---------------|----------------------------------------------|
| Bidirectional | device reads and writes to the memory        |
| To_device     | device reads the memory                      |
| From_device   | device writes to the memory                  |
| None          | device is coherently connected to the memory |

Definition at line 73 of file [dma\\_space](#).

### 14.232.3.3 Space\_attrb

```
enum L4Re::Dma_space::Space_attrb
```

Attributes assigned to the DMA space when associated with a specific device.

See also

[Space\\_attrbs](#)

#### Enumerator

|            |                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Coherent   | The device is connected coherently with the cache. This means that the <a href="#">map()</a> and <a href="#">unmap()</a> do not need to sync CPU caches before and after DMA. |
| Phys_space | The DMA space has no DMA task assigned and uses the CPUs physical memory.                                                                                                     |

Definition at line 113 of file [dma\\_space](#).

## 14.232.4 Member Function Documentation

### 14.232.4.1 associate()

```
long L4Re::Dma_space::associate (
 L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task,
 Space_attrbs attr)
```

Associate a DMA task for a device to this [Dma\\_space](#).

#### Precondition

requires capability rights: {RW}

#### Parameters

|    |                 |                                                                                                                                                                                                                                                               |
|----|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>dma_task</i> | The DMA task used for the device that shall be associated with this DMA space. The <i>dma_task</i> might be an invalid capability when <a href="#">Phys_space</a> is set in <i>attr</i> , in this case the CPUs physical memory is used as DMA address space. |
| in | <i>attr</i>     | Attributes for this DMA space. See <a href="#">Space_attrb</a> .                                                                                                                                                                                              |

## 14.232.4.2 disassociate()

```
long L4Re::Dma_space::disassociate ()
```

Disassociate the DMA task from this [Dma\\_space](#).

**Precondition**

requires capability rights: {RW}

## 14.232.4.3 map()

```
long L4Re::Dma_space::map (
 L4::Ipc::Cap< L4Re::Dataspace > src,
 l4_addr_t offset,
 L4::Ipc::In_out< l4_size_t *> size,
 Attributes attrs,
 Direction dir,
 Dma_addr * dma_addr)
```

Map the given part of this data space into the DMA address space.

**Precondition**

requires capability rights: {R}

**Parameters**

|         |                 |                                                                                                                                                                                                                                                                                                                         |
|---------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>src</i>      | Source data space (that describes the memory). Caller needs write rights to the data space.                                                                                                                                                                                                                             |
| in      | <i>offset</i>   | The offset (bytes) within <i>src</i> .                                                                                                                                                                                                                                                                                  |
| in, out | <i>size</i>     | The size (bytes) of the of the region to be mapped to for DMA, after successful mapping the size returned is the size mapped for DMA as single block, this size might be smaller than the original input size, in this case the caller might call <a href="#">map()</a> again with a new offset and the remaining size. |
| in      | <i>attrs</i>    | The attributes used for this DMA mapping (a combination of <a href="#">Dma_space::Attribute</a> values).                                                                                                                                                                                                                |
| in      | <i>dir</i>      | The direction of the DMA transfer issued with this mapping. The same value must later be passed to <a href="#">unmap()</a> .                                                                                                                                                                                            |
| out     | <i>dma_addr</i> | The DMA address to use for DMA with the associated device.                                                                                                                                                                                                                                                              |

**Returns**

0 in the case of success, a negative error code otherwise.

#### 14.232.4.4 unmap()

```
long L4Re::Dma_space::unmap (
 Dma_addr dma_addr,
 l4_size_t size,
 Attributes attrs,
 Direction dir)
```

Unmap the given part of this data space from the DMA address space.

##### Precondition

requires capability rights: {R}

##### Parameters

|                 |                                                                  |
|-----------------|------------------------------------------------------------------|
| <i>dma_addr</i> | The DMA address (returned by <a href="#">Dma_space::map()</a> ). |
| <i>size</i>     | The size (bytes) of the memory region to unmap.                  |
| <i>attrs</i>    | The attributes for the unmap (currently none).                   |
| <i>dir</i>      | The direction of the finished DMA operation.                     |

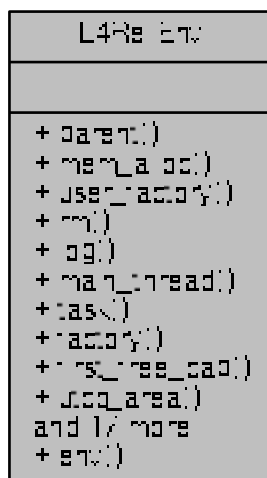
The documentation for this class was generated from the following file:

- [l4/re/dma\\_space](#)

## 14.233 L4Re::Env Class Reference

C++ interface of the initial environment that is provided to an [L4](#) task.

Collaboration diagram for L4Re::Env:





## Public Types

- typedef [l4re\\_env\\_cap\\_entry\\_t](#) [Cap\\_entry](#)  
C++ type for an entry in the initial objects array.

## Public Member Functions

- [L4::Cap](#)< [Parent](#) > [parent](#) () const throw ()  
*Object-capability to the parent.*
- [L4::Cap](#)< [Mem\\_alloc](#) > [mem\\_alloc](#) () const throw ()  
*Object-capability to the memory allocator.*
- [L4::Cap](#)< [L4::Factory](#) > [user\\_factory](#) () const throw ()  
*Object-capability to the user-level object factory.*
- [L4::Cap](#)< [Rm](#) > [rm](#) () const throw ()  
*Object-capability to the region map.*
- [L4::Cap](#)< [Log](#) > [log](#) () const throw ()  
*Object-capability to the logging service.*
- [L4::Cap](#)< [L4::Thread](#) > [main\\_thread](#) () const throw ()  
*Object-capability of the first user thread.*
- [L4::Cap](#)< [L4::Task](#) > [task](#) () const throw ()  
*Object-capability of the user task.*
- [L4::Cap](#)< [L4::Factory](#) > [factory](#) () const throw ()  
*Object-capability to the factory object available to the task.*
- [l4\\_cap\\_idx\\_t](#) [first\\_free\\_cap](#) () const throw ()  
*First available capability selector.*
- [l4\\_fpage\\_t](#) [utcb\\_area](#) () const throw ()  
*UTCB area of the task.*
- [l4\\_addr\\_t](#) [first\\_free\\_utcb](#) () const throw ()  
*First free UTCB.*
- [Cap\\_entry](#) const \* [initial\\_caps](#) () const throw ()  
*Get a pointer to the first entry in the initial objects array.*
- [Cap\\_entry](#) const \* [get](#) (char const \*name, unsigned l) const throw ()  
*Get the Cap\_entry for the object named name.*
- template<typename T >  
[L4::Cap](#)< T > [get\\_cap](#) (char const \*name, unsigned l) const throw ()  
*Get the capability selector for the object named name.*
- template<typename T >  
[L4::Cap](#)< T > [get\\_cap](#) (char const \*name) const throw ()  
*Get the capability selector for the object named name.*
- void [parent](#) ([L4::Cap](#)< [Parent](#) > const &c) throw ()  
*Set parent object-capability.*
- void [mem\\_alloc](#) ([L4::Cap](#)< [Mem\\_alloc](#) > const &c) throw ()  
*Set memory allocator object-capability.*
- void [rm](#) ([L4::Cap](#)< [Rm](#) > const &c) throw ()  
*Set region map object-capability.*
- void [log](#) ([L4::Cap](#)< [Log](#) > const &c) throw ()  
*Set log object-capability.*
- void [main\\_thread](#) ([L4::Cap](#)< [L4::Thread](#) > const &c) throw ()  
*Set object-capability of first user thread.*
- void [factory](#) ([L4::Cap](#)< [L4::Factory](#) > const &c) throw ()

- *Set factory object-capability.*  
void [first\\_free\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) throw ()
- *Set first available capability selector.*  
void [utcb\\_area](#) ([l4\\_fpage\\_t](#) utcb) throw ()
- *Set UTCB area of the task.*  
void [first\\_free\\_utcb](#) ([l4\\_addr\\_t](#) u) throw ()
- *Set first free UTCB.*  
[L4::Cap](#)< [L4::Scheduler](#) > [scheduler](#) () const throw ()
- *Get the scheduler capability for the task.*  
void [scheduler](#) ([L4::Cap](#)< [L4::Scheduler](#) > const &c) throw ()
- *Set the scheduler capability.*  
void [initial\\_caps](#) ([Cap\\_entry](#) \*first) throw ()
- *Set the pointer to the first Cap\_entry in the initial objects array.*

## Static Public Member Functions

- static [Env](#) const \* [env](#) () throw ()  
*Returns the initial environment for the current task.*

### 14.233.1 Detailed Description

C++ interface of the initial environment that is provided to an [L4](#) task.

The initial environment is provided to each [L4](#) task that is started by an [L4Re](#) conform loader, such as the Moe root task. The initial environment provides access to a set of initial capabilities and some additional information about the available resources, such as free UTCBs (see [Virtual Registers](#) ) and available entries in capability table (provided by the micro kernel).

Each of the initial capabilities is stored at a fixed index in the task's capability table and the [L4](#) runtime environment provides convenience functions to retrieve the capabilities. See the table below for an comprehensive overview.

| Name         | Object Type                   | Convenience Function                      |
|--------------|-------------------------------|-------------------------------------------|
| parent       | <a href="#">L4Re::Parent</a>  | <a href="#">L4Re::Env::parent()</a>       |
| user_factory | <a href="#">L4::Factory</a>   | <a href="#">L4Re::Env::user_factory()</a> |
| log          | <a href="#">L4Re::Log</a>     | <a href="#">L4Re::Env::log()</a>          |
| main_thread  | <a href="#">L4::Thread</a>    | <a href="#">L4Re::Env::main_thread()</a>  |
| rm           | <a href="#">L4Re::Rm</a>      | <a href="#">L4Re::Env::rm()</a>           |
| factory      | <a href="#">L4::Factory</a>   | <a href="#">L4Re::Env::factory()</a>      |
| task         | <a href="#">L4::Task</a>      | <a href="#">L4Re::Env::task()</a>         |
| scheduler    | <a href="#">L4::Scheduler</a> | <a href="#">L4Re::Env::scheduler()</a>    |

Additional information found in the initial environment is:

- First free entry in capability table
- The [UTCB](#) area (as flex page)
- First free UTCB (address in the UTCB area)

## Include File

```
#include <l4/re/env>
```

For an explanation of the default task capabilities see [l4\\_default\\_caps\\_t](#).

For the C interface refer to [Initial Environment](#).

Definition at line 85 of file [env](#).

## 14.233.2 Member Function Documentation

## 14.233.2.1 env()

```
static Env const* L4Re::Env::env () throw () [inline], [static]
```

Returns the initial environment for the current task.

## Returns

Pointer to the initial environment class.

A typical use of this function is [L4Re::Env::env\(\)](#)-><member>()

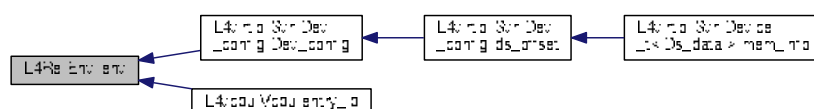
## Examples:

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 103 of file [env](#).

Referenced by [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#), and [L4vcpu::Vcpu::entry\\_ip\(\)](#).

Here is the caller graph for this function:



**14.233.2.2** `factory()` [1/2]

```
L4::Cap<L4::Factory> L4Re::Env::factory () const throw () [inline]
```

Object-capability to the factory object available to the task.

**Returns**

Factory object-capability

Definition at line 151 of file [env](#).

References [l4re\\_env\\_t::factory](#).

**14.233.2.3** `factory()` [2/2]

```
void L4Re::Env::factory (
 L4::Cap< L4::Factory > const & c) throw () [inline]
```

Set factory object-capability.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <code>c</code> | Factory object-capability |
|----------------|---------------------------|

Definition at line 256 of file [env](#).

References [l4re\\_env\\_t::factory](#).

**14.233.2.4** `first_free_cap()` [1/2]

```
l4_cap_idx_t L4Re::Env::first_free_cap () const throw () [inline]
```

First available capability selector.

**Returns**

First capability selector.

First capability selector available for use for in the application.

Definition at line 159 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_cap](#).

**14.233.2.5** `first_free_cap()` [2/2]

```
void L4Re::Env::first_free_cap (
 l4_cap_idx_t c) throw () [inline]
```

Set first available capability selector.

## Parameters

|                |                                                         |
|----------------|---------------------------------------------------------|
| <code>c</code> | First capability selector available to the application. |
|----------------|---------------------------------------------------------|

Definition at line 262 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_cap](#).

## 14.233.2.6 first\_free\_utcb() [1/2]

```
l4_addr_t L4Re::Env::first_free_utcb () const throw () [inline]
```

First free UTCB.

## Returns

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 174 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_utcb](#).

## 14.233.2.7 first\_free\_utcb() [2/2]

```
void L4Re::Env::first_free_utcb (
 l4_addr_t u) throw () [inline]
```

Set first free UTCB.

## Parameters

|                |                                                  |
|----------------|--------------------------------------------------|
| <code>u</code> | First UTCB available for the application to use. |
|----------------|--------------------------------------------------|

Definition at line 274 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_utcb](#).

## 14.233.2.8 get()

```
Cap_entry const* L4Re::Env::get (
 char const * name,
 unsigned l) const throw () [inline]
```

Get the `Cap_entry` for the object named *name*.

**Parameters**

|             |                                                                           |
|-------------|---------------------------------------------------------------------------|
| <i>name</i> | is the name of the object.                                                |
| <i>l</i>    | is the length of the name, thus <i>name</i> might not be zero terminated. |

**Returns**

A pointer to the `Cap_entry` for the object named *name*, or NULL if no such object was found.

Definition at line 192 of file [env](#).

References [l4re\\_env\\_get\\_cap\\_l\(\)](#).

Here is the call graph for this function:

**14.233.2.9 get\_cap()** [1/2]

```

template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
 char const * name,
 unsigned l) const throw () [inline]

```

Get the capability selector for the object named *name*.

**Parameters**

|             |                                                                           |
|-------------|---------------------------------------------------------------------------|
| <i>name</i> | is the name of the object.                                                |
| <i>l</i>    | is the length of the name, thus <i>name</i> might not be zero terminated. |

**Returns**

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

**Examples:**

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 204 of file [env](#).

References [L4\\_ENOENT](#).

#### 14.233.2.10 `get_cap()` [2/2]

```
template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
 char const * name) const throw () [inline]
```

Get the capability selector for the object named *name*.

##### Parameters

|             |                                              |
|-------------|----------------------------------------------|
| <i>name</i> | is the name of the object (zero terminated). |
|-------------|----------------------------------------------|

##### Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Definition at line 219 of file [env](#).

#### 14.233.2.11 `initial_caps()` [1/2]

```
Cap_entry const* L4Re::Env::initial_caps () const throw () [inline]
```

Get a pointer to the first entry in the initial objects array.

##### Returns

A pointer to the first entry in the initial objects array.

Definition at line 181 of file [env](#).

#### 14.233.2.12 `initial_caps()` [2/2]

```
void L4Re::Env::initial_caps (
 Cap_entry * first) throw () [inline]
```

Set the pointer to the first `Cap_entry` in the initial objects array.

## Parameters

|              |                                    |
|--------------|------------------------------------|
| <i>first</i> | is the first element in the array. |
|--------------|------------------------------------|

Definition at line 296 of file [env](#).

14.233.2.13 `log()` [1/2]

```
L4::Cap<Log> L4Re::Env::log () const throw () [inline]
```

Object-capability to the logging service.

## Returns

[Log](#) object-capability

Definition at line 133 of file [env](#).

References [l4re\\_env\\_t::log](#).

14.233.2.14 `log()` [2/2]

```
void L4Re::Env::log (
 L4::Cap< Log > const & c) throw () [inline]
```

Set log object-capability.

## Parameters

|          |                                       |
|----------|---------------------------------------|
| <i>c</i> | <a href="#">Log</a> object-capability |
|----------|---------------------------------------|

Definition at line 244 of file [env](#).

References [l4re\\_env\\_t::log](#).

14.233.2.15 `main_thread()` [1/2]

```
L4::Cap<L4::Thread> L4Re::Env::main_thread () const throw () [inline]
```

Object-capability of the first user thread.



**Returns**

Object-capability of the first user thread.

Definition at line 139 of file [env](#).

References [l4re\\_env\\_t::main\\_thread](#).

**14.233.2.16 main\_thread()** [2/2]

```
void L4Re::Env::main_thread (
 L4::Cap< L4::Thread > const & c) throw () [inline]
```

Set object-capability of first user thread.

**Parameters**

|          |                                  |
|----------|----------------------------------|
| <b>c</b> | First thread's object-capability |
|----------|----------------------------------|

Definition at line 250 of file [env](#).

References [l4re\\_env\\_t::main\\_thread](#).

**14.233.2.17 mem\_alloc()** [1/2]

```
L4::Cap<Mem_alloc> L4Re::Env::mem_alloc () const throw () [inline]
```

Object-capability to the memory allocator.

**Returns**

Memory allocator object-capability

**Examples:**

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 116 of file [env](#).

References [l4re\\_env\\_t::mem\\_alloc](#).

**14.233.2.18 mem\_alloc()** [2/2]

```
void L4Re::Env::mem_alloc (
 L4::Cap< Mem_alloc > const & c) throw () [inline]
```

Set memory allocator object-capability.

## Parameters

|                |                                    |
|----------------|------------------------------------|
| <code>c</code> | Memory allocator object-capability |
|----------------|------------------------------------|

Definition at line 232 of file [env](#).

References [l4re\\_env\\_t::mem\\_alloc](#).

14.233.2.19 `parent()` [1/2]

```
L4::Cap<Parent> L4Re::Env::parent () const throw () [inline]
```

Object-capability to the parent.

## Returns

[Parent](#) object-capability

Definition at line 110 of file [env](#).

References [l4re\\_env\\_t::parent](#).

14.233.2.20 `parent()` [2/2]

```
void L4Re::Env::parent (
 L4::Cap< Parent > const & c) throw () [inline]
```

Set parent object-capability.

## Parameters

|                |                                          |
|----------------|------------------------------------------|
| <code>c</code> | <a href="#">Parent</a> object-capability |
|----------------|------------------------------------------|

Definition at line 226 of file [env](#).

References [l4re\\_env\\_t::parent](#).

14.233.2.21 `rm()` [1/2]

```
L4::Cap<Rm> L4Re::Env::rm () const throw () [inline]
```

Object-capability to the region map.

**Returns**

Region map object-capability

**Examples:**

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 127 of file [env](#).

References [l4re\\_env\\_t::rm](#).

**14.233.2.22 rm()** [2/2]

```
void L4Re::Env::rm (
 L4::Cap< Rm > const & c) throw () [inline]
```

Set region map object-capability.

**Parameters**

|          |                              |
|----------|------------------------------|
| <b>c</b> | Region map object-capability |
|----------|------------------------------|

Definition at line 238 of file [env](#).

References [l4re\\_env\\_t::rm](#).

**14.233.2.23 scheduler()** [1/2]

```
L4::Cap<L4::Scheduler> L4Re::Env::scheduler () const throw () [inline]
```

Get the scheduler capability for the task.

**Returns**

The capability selector for the default scheduler used for this task.

**Examples:**

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 282 of file [env](#).

References [l4re\\_env\\_t::scheduler](#).

**14.233.2.24 scheduler()** [2/2]

```
void L4Re::Env::scheduler (
 L4::Cap< L4::Scheduler > const & c) throw () [inline]
```

Set the scheduler capability.

## Parameters

|                |                                           |
|----------------|-------------------------------------------|
| <code>c</code> | is the capability to be set as scheduler. |
|----------------|-------------------------------------------|

Definition at line 289 of file [env](#).

References [l4re\\_env\\_t::scheduler](#).

## 14.233.2.25 task()

```
L4::Cap<L4::Task> L4Re::Env::task () const throw () [inline]
```

Object-capability of the user task.

## Returns

Object-capability of the user task.

Definition at line 145 of file [env](#).

## 14.233.2.26 utcb\_area() [1/2]

```
l4_fpage_t L4Re::Env::utcb_area () const throw () [inline]
```

UTCB area of the task.

## Returns

UTCB area

Definition at line 165 of file [env](#).

References [l4re\\_env\\_t::utcb\\_area](#).

## 14.233.2.27 utcb\_area() [2/2]

```
void L4Re::Env::utcb_area (
 l4_fpage_t utcbs) throw () [inline]
```

Set UTCB area of the task.

## Parameters

|               |           |
|---------------|-----------|
| <i>utcb</i> s | UTCB area |
|---------------|-----------|

Definition at line 268 of file [env](#).

References [l4re\\_env\\_t::utcb\\_area](#).

The documentation for this class was generated from the following file:

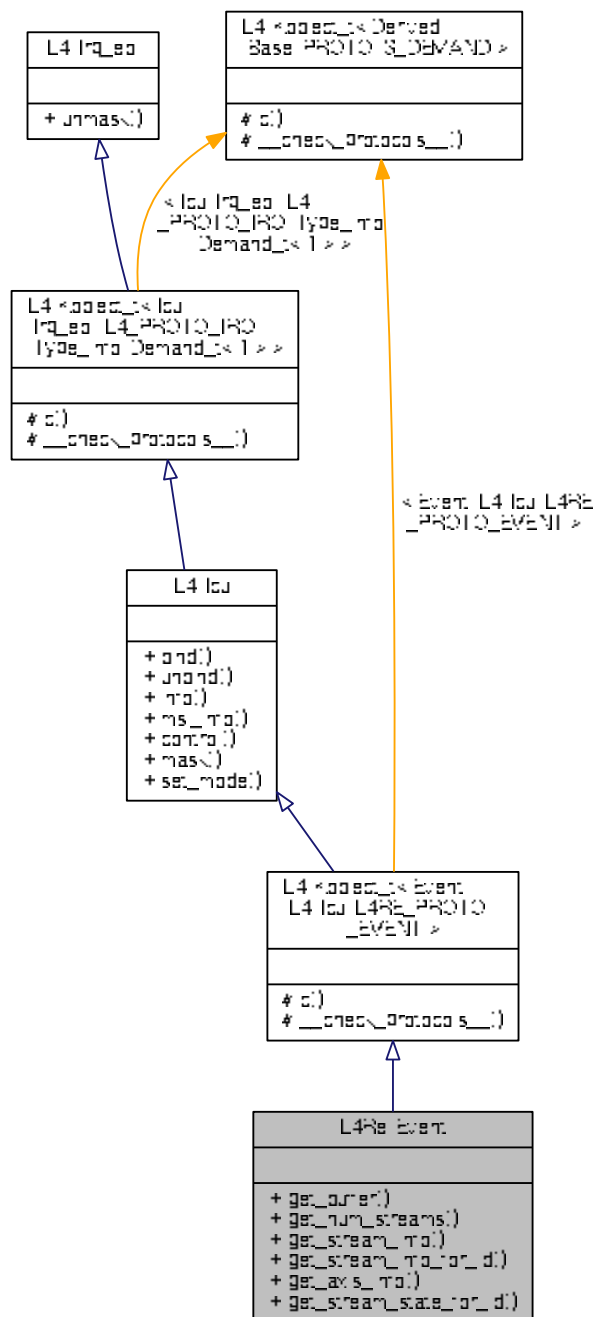
- [l4/re/env](#)

## 14.234 L4Re::Event Class Reference

[Event](#) class.



Collaboration diagram for L4Re::Event:



## Public Member Functions

- long `get_buffer` (L4::ipc::Out< L4::Cap< Dataspace > > ds)  
*Get event signal buffer.*

## Additional Inherited Members

### 14.234.1 Detailed Description

[Event](#) class.

See also

[L4Re Event API](#)

Definition at line 144 of file [event](#).

### 14.234.2 Member Function Documentation

#### 14.234.2.1 `get_buffer()`

```
long L4Re::Event::get_buffer (
 L4::Ipc::Out< L4::Cap< Dataspace > > ds)
```

Get event signal buffer.

Return values

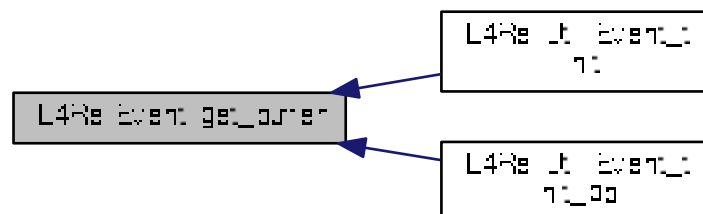
|           |                               |
|-----------|-------------------------------|
| <i>ds</i> | <a href="#">Event</a> buffer. |
|-----------|-------------------------------|

Returns

0 on success, negative error code otherwise.

Referenced by [L4Re::Util::Event\\_t< PAYLOAD >::init\(\)](#), and [L4Re::Util::Event\\_t< PAYLOAD >::init\\_poll\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

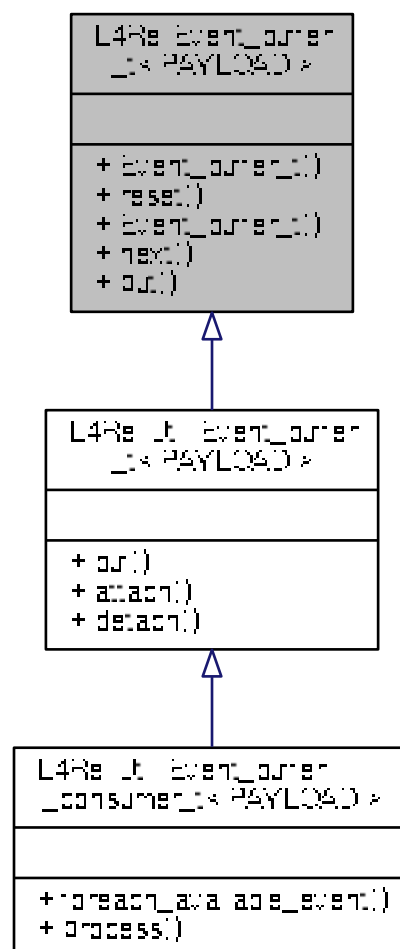


- [l4/re/event](#)

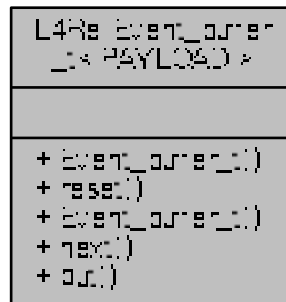
## 14.235 L4Re::Event\_buffer\_t< PAYLOAD > Class Template Reference

[Event](#) buffer class.

Inheritance diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



## Data Structures

- struct [Event](#)  
*Event structure used in buffer.*

## Public Member Functions

- [Event\\_buffer\\_t](#) (void \*buffer, [l4\\_addr\\_t](#) size)  
*Initialize event buffer.*
- [Event](#) \* [next](#) () throw ()  
*Next event in buffer.*
- bool [put](#) ([Event](#) const &ev) throw ()  
*Put event into buffer at current position.*

### 14.235.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line [198](#) of file [event](#).

### 14.235.2 Constructor & Destructor Documentation

#### 14.235.2.1 Event\_buffer\_t()

```
template<typename PAYLOAD = Default_event_payload>
L4Re::Event_buffer_t< PAYLOAD >::Event_buffer_t (
 void * buffer,
 l4_addr_t size) [inline]
```

Initialize event buffer.

## Parameters

|               |                          |
|---------------|--------------------------|
| <i>buffer</i> | Pointer to buffer.       |
| <i>size</i>   | Size of buffer in bytes. |

Definition at line 245 of file [event](#).

### 14.235.3 Member Function Documentation

#### 14.235.3.1 next()

```
template<typename PAYLOAD = Default_event_payload>
Event* L4Re::Event_buffer_t< PAYLOAD >::next () throw () [inline]
```

Next event in buffer.

## Returns

0 if no event available, event otherwise.

Definition at line 255 of file [event](#).

References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

#### 14.235.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
 Event const & ev) throw () [inline]
```

Put event into buffer at current position.

## Parameters

|           |                                               |
|-----------|-----------------------------------------------|
| <i>ev</i> | <a href="#">Event</a> to put into the buffer. |
|-----------|-----------------------------------------------|

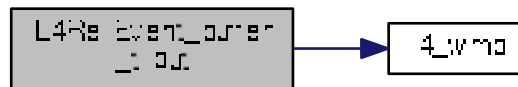
## Returns

false if buffer is full and entry could not be added.

Definition at line 272 of file [event](#).

References [l4\\_wmb\(\)](#), and [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

Here is the call graph for this function:



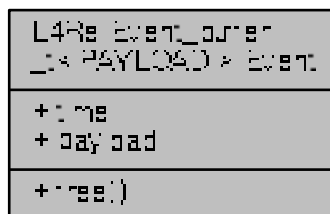
The documentation for this class was generated from the following file:

- l4/re/event

## 14.236 L4Re::Event\_buffer\_t< PAYLOAD >::Event Struct Reference

[Event](#) structure used in buffer.

Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >::Event:



### Public Member Functions

- void [free](#) () throw ()  
*Free the entry.*

### Data Fields

- long long [time](#)  
*[Event](#) time stamp.*

### 14.236.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
struct L4Re::Event_buffer_t< PAYLOAD >::Event
```

[Event](#) structure used in buffer.

Definition at line 205 of file [event](#).

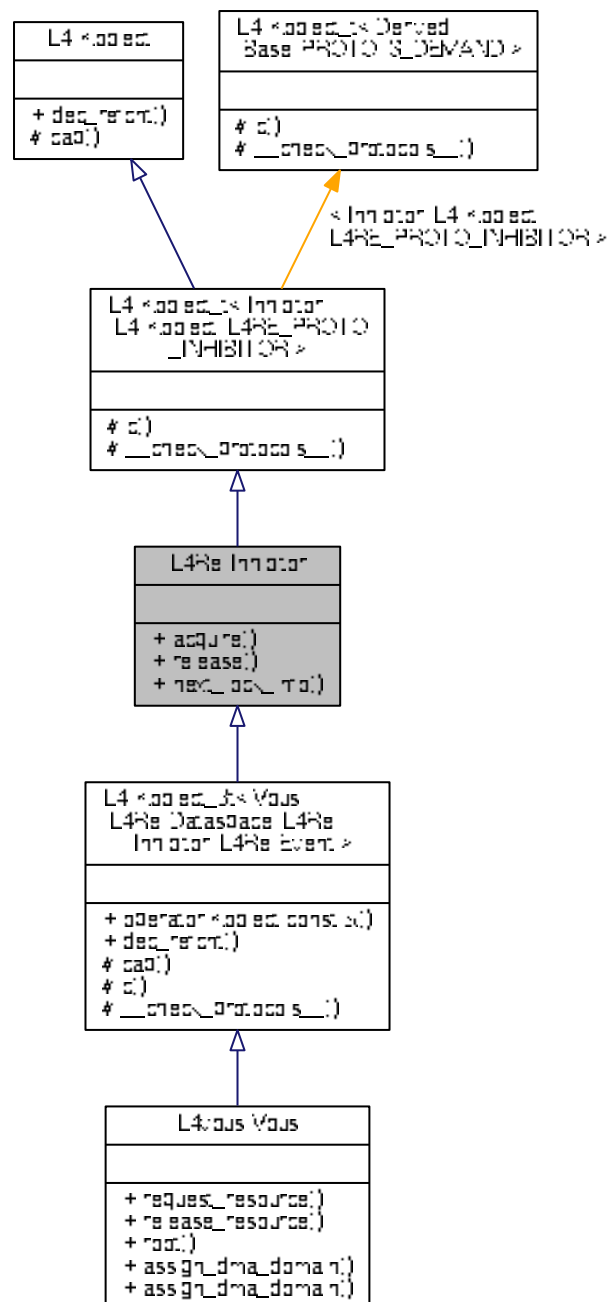
The documentation for this struct was generated from the following file:

- [l4/re/event](#)

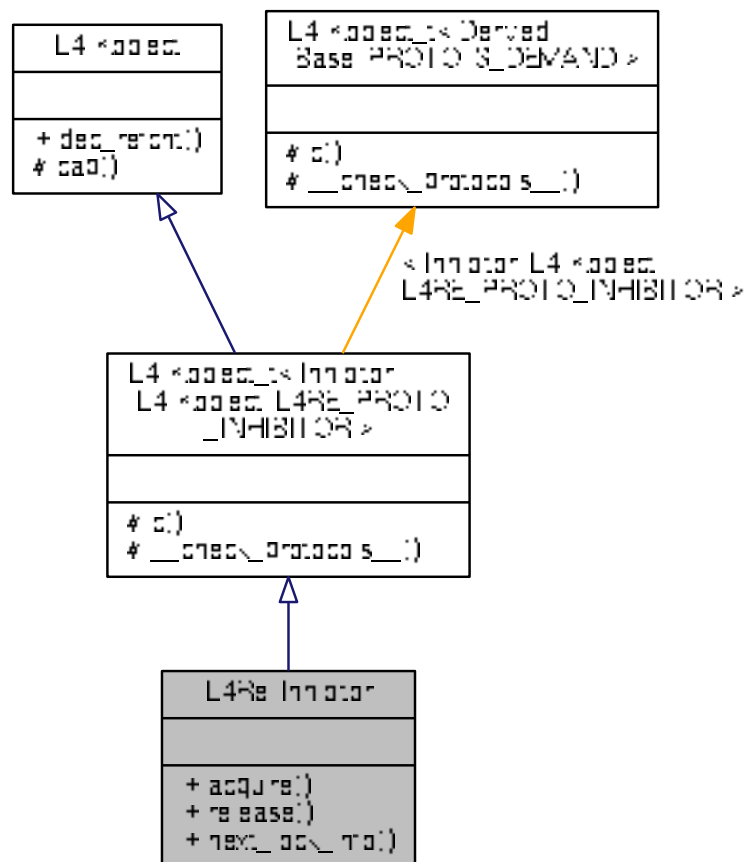
## 14.237 L4Re::Inhibitor Class Reference

Set of inhibitor locks, which inhibit specific actions when held.

Inheritance diagram for L4Re::Inhibitor:



Collaboration diagram for L4Re::Inhibitor:



## Public Types

- enum { `Name_max` = 20 }

## Public Member Functions

- long `acquire` (`l4_umword_t` id, `L4::lpc::String<>` reason)  
*Acquire a specific inhibitor lock.*
- long `release` (`l4_umword_t` id)  
*Release a specific inhibitor lock.*
- long `next_lock_info` (char \*name, unsigned len, `l4_mword_t` current\_id=-1, `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Get information for the next available inhibitor lock.*

## Additional Inherited Members

### 14.237.1 Detailed Description

Set of inhibitor locks, which inhibit specific actions when held.

This interface provides access to a set of inhibitor locks, each determined by an ID that is specific to the [Inhibitor](#) object. Each individual lock shall prevent, a specific (implementation defined) action to be executed, as long as the lock is held.

For example there can be an inhibitor lock to prevent a transition to suspend-to-RAM state and a different one to prevent shutdown.

A client shall take an inhibitor lock if it needs to execute code before the action is taken. For example a lock-screen application shall grab an inhibitor lock for the suspend action to be able to lock the screen before the system goes to sleep.

[Inhibitor](#) locks are usually closely related to specific events. Usually a server automatically subscribes a client holding a lock to the corresponding event. The server shall send the event to inform the client that an action is pending. Upon reception of the event, the client is supposed to release the corresponding inhibitor lock.

Definition at line 40 of file [inhibitor](#).

### 14.237.2 Member Enumeration Documentation

#### 14.237.2.1 anonymous enum

anonymous enum

##### Enumerator

|          |                                      |
|----------|--------------------------------------|
| Name_max | The maximum length of a lock's name. |
|----------|--------------------------------------|

Definition at line 44 of file [inhibitor](#).

### 14.237.3 Member Function Documentation

#### 14.237.3.1 acquire()

```
long L4Re::Inhibitor::acquire (
 l4_umword_t id,
 L4::Ipc::String<> reason)
```

Acquire a specific inhibitor lock.



## Parameters

|               |                                                                             |
|---------------|-----------------------------------------------------------------------------|
| <i>id</i>     | ID of the inhibitor lock that the client intends to acquire                 |
| <i>reason</i> | The reason why you need the lock. Used for informing the user or debugging. |

## Return values

|            |                                         |
|------------|-----------------------------------------|
| 0          | Success                                 |
| -L4_ENODEV | The specified <i>id</i> does not exist. |

## 14.237.3.2 next\_lock\_info()

```
long L4Re::Inhibitor::next_lock_info (
 char * name,
 unsigned len,
 l4_mword_t current_id = -1,
 l4_utcb_t * utcb = l4_utcb()) [inline]
```

Get information for the next available inhibitor lock.

## Parameters

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <i>name</i>       | A pointer to a buffer for the name of the lock.                               |
| <i>len</i>        | The length of the available buffer (usually <a href="#">Name_max</a> is used) |
| <i>current_id</i> | The ID of the last available lock, use -1 to get the first lock.              |
| <i>utcb</i>       | The UTCB to use for the message.                                              |

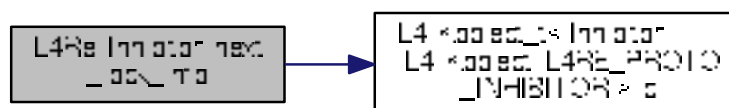
## Return values

|            |                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------|
| >0         | The ID of the next available lock if there is one (in this case <i>name</i> shall contain the name of the inhibitor lock). |
| -L4_ENODEV | There are no more locks.                                                                                                   |

Definition at line 86 of file [inhibitor](#).

References [L4::Kobject\\_t< Inhibitor](#), [L4::Kobject](#), [L4RE\\_PROTO\\_INHIBITOR >::c\(\)](#), and [L4\\_INLINE\\_RPC\\_NF](#).

Here is the call graph for this function:



### 14.237.3.3 release()

```
long L4Re::Inhibitor::release (
 l4_umword_t id)
```

Release a specific inhibitor lock.

#### Parameters

|           |                                          |
|-----------|------------------------------------------|
| <i>id</i> | The ID of the inhibitor lock to release. |
|-----------|------------------------------------------|

#### Return values

|            |                                               |
|------------|-----------------------------------------------|
| 0          | Success                                       |
| -L4_ENODEV | Lock with the given <i>id</i> does not exist. |

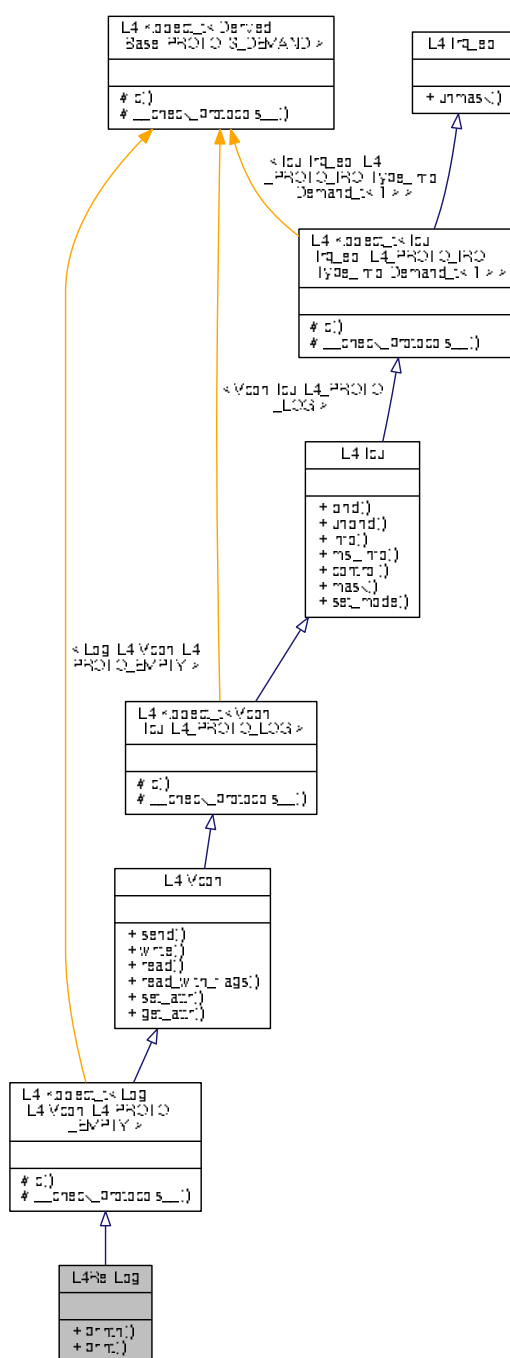
The documentation for this class was generated from the following file:

- l4/re/inhibitor

## 14.238 L4Re::Log Class Reference

[Log](#) interface class.

Inheritance diagram for L4Re::Log:





- Generated for L4Re by Doxygen

## Additional Inherited Members

### 14.238.1 Detailed Description

[Log](#) interface class.

Definition at line 44 of file [log](#).

### 14.238.2 Member Function Documentation

#### 14.238.2.1 print()

```
void L4Re::Log::print (
 char const * string) const throw)
```

Print NULL-terminated string.

##### Parameters

|               |                 |
|---------------|-----------------|
| <i>string</i> | string to print |
|---------------|-----------------|

#### 14.238.2.2 printn()

```
void L4Re::Log::printn (
 char const * string,
 int len) const throw)
```

Print string with length len, NULL characters don't matter.

##### Parameters

|               |                  |
|---------------|------------------|
| <i>string</i> | string to print  |
| <i>len</i>    | length of string |

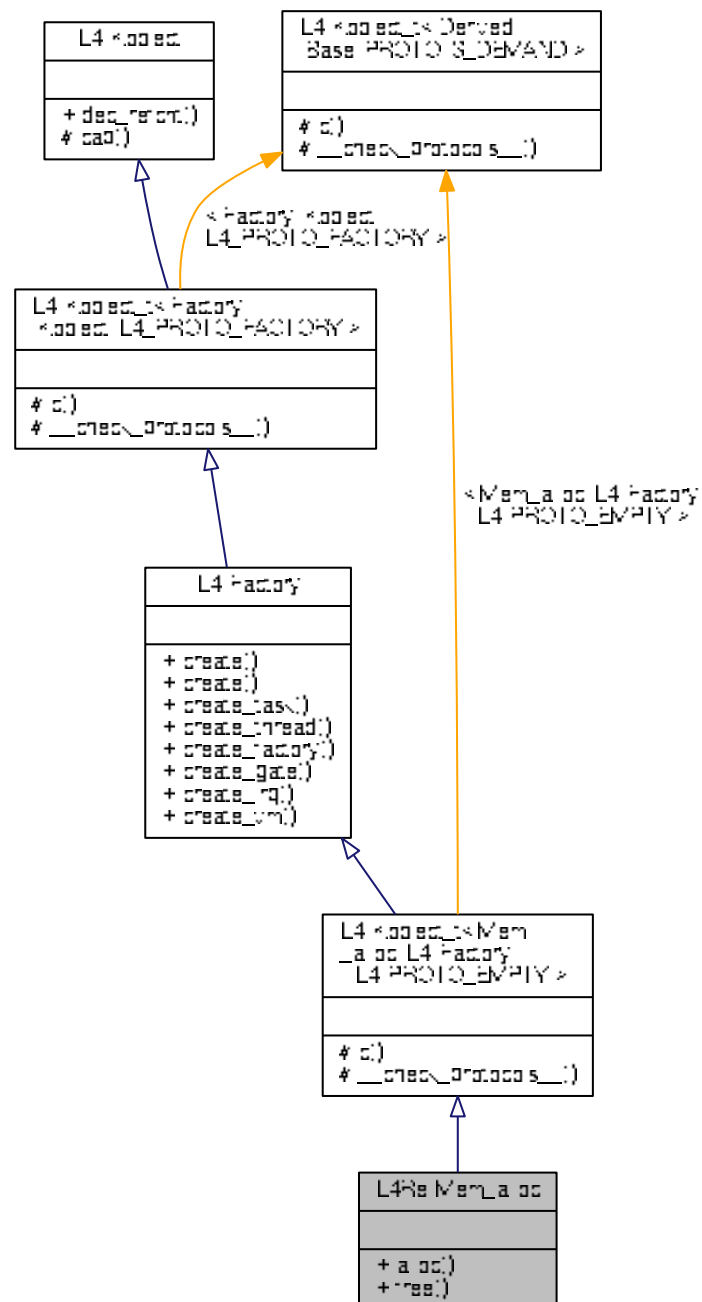
The documentation for this class was generated from the following file:

- [l4/re/log](#)

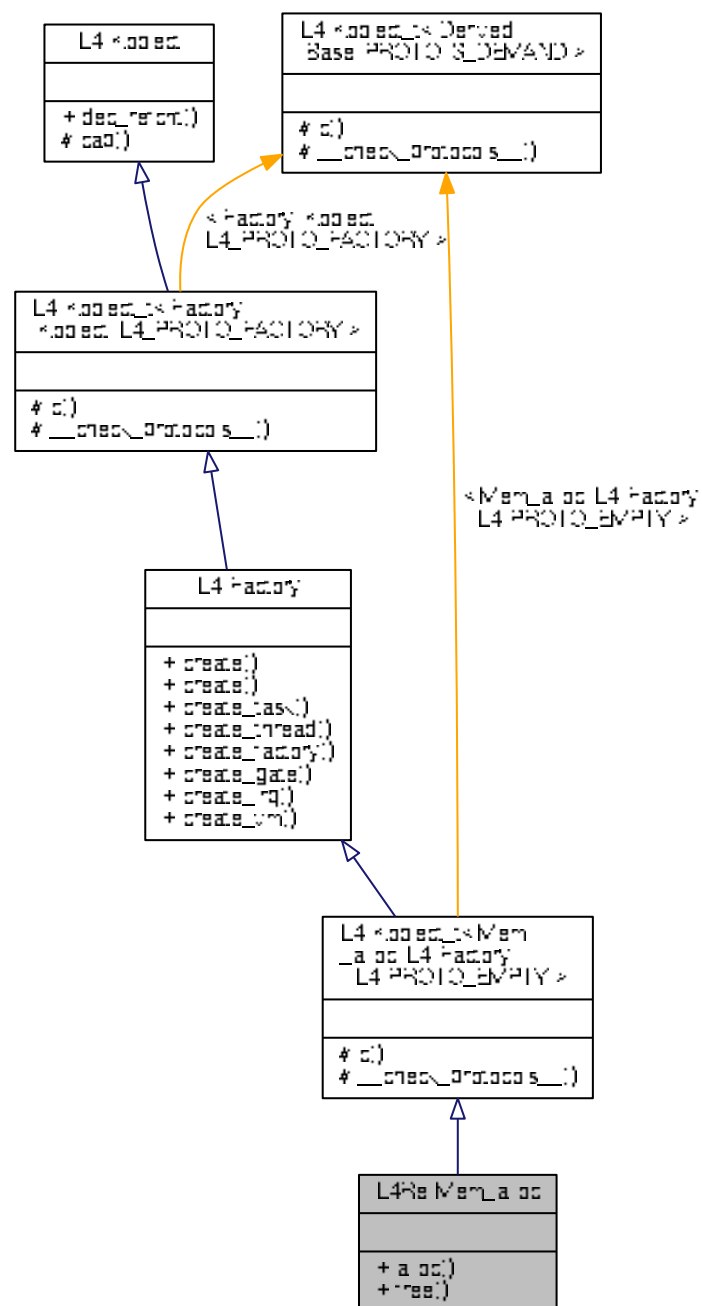
## 14.239 L4Re::Mem\_alloc Class Reference

Memory allocation interface.

Inheritance diagram for L4Re::Mem\_alloc:



Collaboration diagram for L4Re::Mem\_alloc:



## Public Types

- enum `Mem_alloc_flags` { `Continuous` = 0x01, `Pinned` = 0x02, `Super_pages` = 0x04 }

*Flags for the allocator.*

## Public Member Functions

- long `alloc` (long size, `L4::Cap< Dataspace >` mem, unsigned long flags=0, unsigned long align=0) const throw ()  
*Allocate anonymous memory.*
- long `free` (`L4::Cap< Dataspace >` mem) const throw ()  
*Free dataspace.*

## Additional Inherited Members

### 14.239.1 Detailed Description

Memory allocation interface.

The memory-allocator API is the basic API to allocate memory from the `L4Re` subsystem. The memory is allocated in terms of dataspace (see `L4Re::Dataspace`). The provided dataspace have at least the property that data written to such a dataspace is available as long as the dataspace is not freed or the data is not overwritten. In particular, the memory backing a dataspace from an allocator need not be allocated instantly, but may be allocated lazily on demand.

A memory allocator can provide dataspace with additional properties, such as physically contiguous memory, pre-allocated memory, or pinned memory. To request memory with an additional property the `L4Re::Mem_alloc::alloc()` method provides a flags parameter. If the concrete implementation of a memory allocator does not support or allow allocation of memory with a certain property, the allocation may be refused.

Definition at line 61 of file `mem_alloc`.

### 14.239.2 Member Enumeration Documentation

#### 14.239.2.1 Mem\_alloc\_flags

```
enum L4Re::Mem_alloc::Mem_alloc_flags
```

Flags for the allocator.

They describe requested properties of the allocated memory. Support of these properties by the dataspace provider is optional.

#### Enumerator

|             |                                                       |
|-------------|-------------------------------------------------------|
| Continuous  | Allocate physically contiguous memory.                |
| Pinned      | Deprecated, use <code>L4Re::Dma_space</code> instead. |
| Super_pages | Allocate super pages.                                 |

Definition at line 71 of file `mem_alloc`.



### 14.239.3 Member Function Documentation

#### 14.239.3.1 alloc()

```
long L4Re::Mem_alloc::alloc (
 long size,
 L4::Cap< Dataspace > mem,
 unsigned long flags = 0,
 unsigned long align = 0) const throw ()
```

Allocate anonymous memory.

##### Parameters

|     |              |                                                                                                                                                                                                                                                                                                                                                                     |
|-----|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>size</i>  | Size in bytes to be requested (granularity is (super)pages and the size is rounded up to this granularity). If <i>size</i> is a negative value and <i>flags</i> set the <code>Mem_alloc_flags::Continuous</code> bit the allocator tries to allocate as much memory as possible leaving an amount of at least <code>-size</code> bytes within the associated quota. |
| out | <i>mem</i>   | Capability slot where the capability to the dataspace is received.                                                                                                                                                                                                                                                                                                  |
|     | <i>flags</i> | Special dataspace properties, see <a href="#">Mem_alloc_flags</a>                                                                                                                                                                                                                                                                                                   |
|     | <i>align</i> | Log2 alignment of dataspace if supported by allocator, will be at least <code>L4_PAGESHIFT</code> , with <code>Super_pages</code> flag set at least <code>L4_SUPERPAGESHIFT</code>                                                                                                                                                                                  |

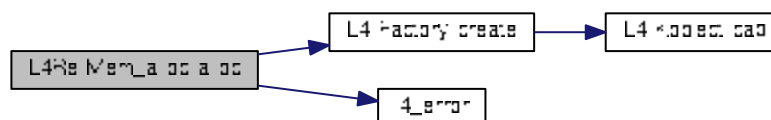
##### Return values

|            |                              |
|------------|------------------------------|
| 0          | Success                      |
| -L4_ERANGE | Given size not supported.    |
| -L4_ENOMEM | Not enough memory available. |
| <0         | IPC error                    |

Definition at line 35 of file [mem\\_alloc\\_impl.h](#).

References [L4::Factory::create\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



## 14.239.3.2 free()

```
long L4Re::Mem_alloc::free (
 L4::Cap< Dataspace > mem) const throw ()
```

Free dataspace.

## Parameters

|            |                           |
|------------|---------------------------|
| <i>mem</i> | Dataspace to be released. |
|------------|---------------------------|

## Return values

|   |  |
|---|--|
| 0 |  |
|---|--|

**Deprecated** This function is an empty stub which remains here for backward compatibility only. Use `L4::Task↵::unmap(mem, L4_FP_DELETE_OBJ)` or other similar means to remove a dataspace.

Definition at line 47 of file [mem\\_alloc\\_impl.h](#).

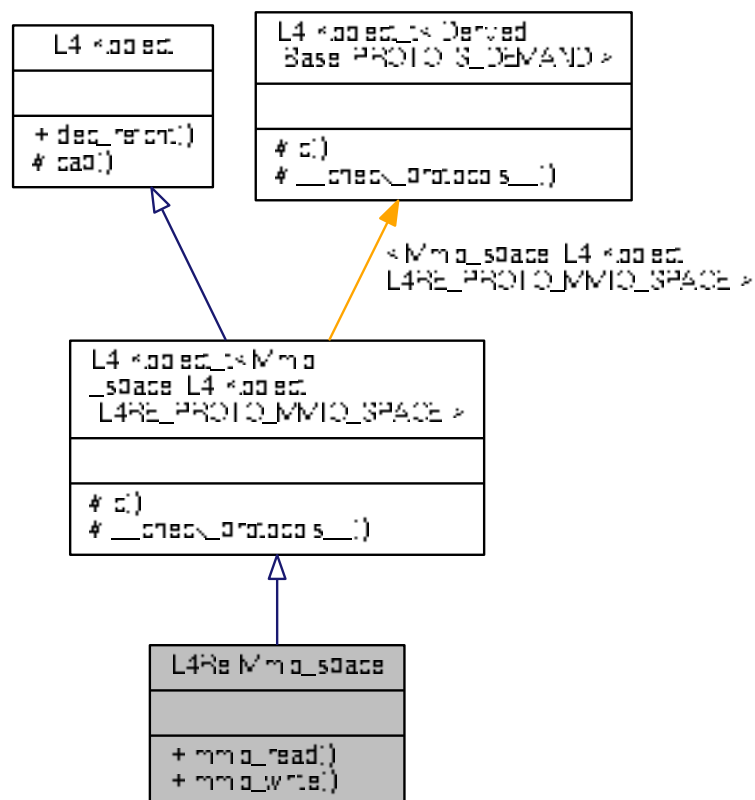
The documentation for this class was generated from the following files:

- [l4/re/mem\\_alloc](#)
- [l4/re/impl/mem\\_alloc\\_impl.h](#)

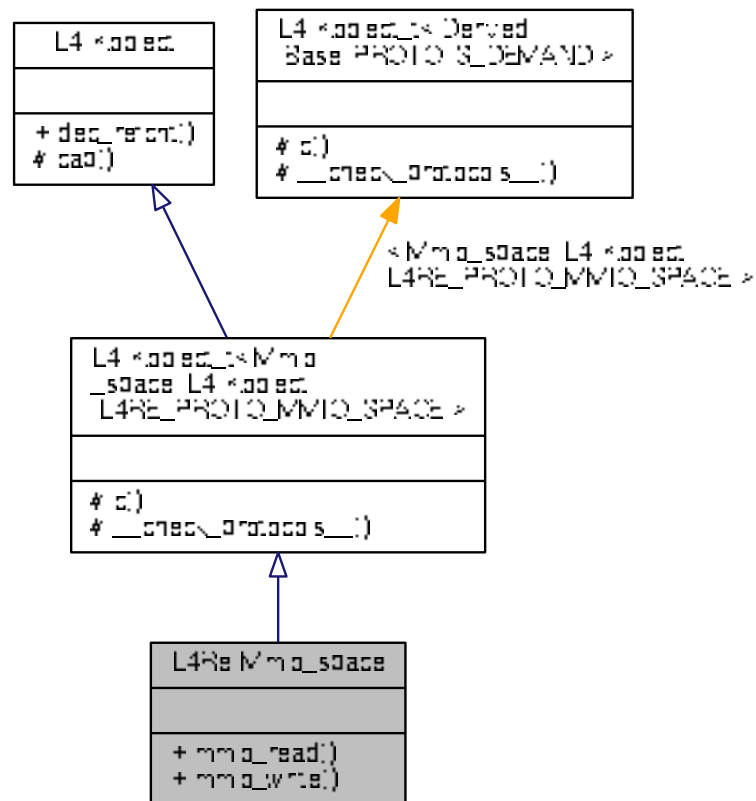
## 14.240 L4Re::Mmio\_space Struct Reference

Interface for memory-like address space accessible via IPC.

Inheritance diagram for L4Re::Mmio\_space:



Collaboration diagram for L4Re::Mmio\_space:



## Public Types

- enum `Access_width` { `Wd_8bit` = 0, `Wd_16bit` = 1, `Wd_32bit` = 2, `Wd_64bit` = 3 }
- Actual size of the value to read or write.*
- typedef `l4_uint64_t Addr`
- Device address.*

## Public Member Functions

- long `mmio_read` (`Addr addr`, char width, `l4_uint64_t *value`)
- Read a value from the given address.*
- long `mmio_write` (`Addr addr`, char width, `l4_uint64_t value`)
- Write a value to the given address.*

## Additional Inherited Members

### 14.240.1 Detailed Description

Interface for memory-like address space accessible via IPC.

This interface defines methods for indirect access to MMIO regions.

Memory mapped IO (MMIO) is used by device drivers to control hardware devices. Access to MMIO regions is assigned to user-level device drivers via mappings of memory pages.

However, there are hardware platforms where MMIO regions for different devices share the same memory page. With respect to security and safety, it is often not allowed to map a memory page to multiple device drivers because the driver of one device could then influence operation of another device, which violates security boundaries.

A solution to that problem is to implement a third (trusted) component that gets exclusive access to the shared memory page, and that drivers can access via IPC with the [Mmio\\_space](#) protocol. This proxy-component can then enforce an access policy.

#### Include File

```
#include <l4/re/mmio_space>
```

Definition at line 42 of file [mmio\\_space](#).

### 14.240.2 Member Enumeration Documentation

#### 14.240.2.1 Access\_width

```
enum L4Re::Mmio_space::Access_width
```

Actual size of the value to read or write.

#### Enumerator

|          |                         |
|----------|-------------------------|
| Wd_8bit  | Value is a byte.        |
| Wd_16bit | Value is a 2-byte word. |
| Wd_32bit | Value is a 4-byte word. |
| Wd_64bit | Value is a 8-byte word. |

Definition at line 46 of file [mmio\\_space](#).

### 14.240.3 Member Function Documentation

### 14.240.3.1 mmio\_read()

```
long L4Re::Mmio_space::mmio_read (
 Addr addr,
 char width,
 l4_uint64_t * value)
```

Read a value from the given address.

#### Parameters

|     |              |                                                                                                |
|-----|--------------|------------------------------------------------------------------------------------------------|
|     | <i>addr</i>  | Device virtual address to read from. The address must be aligned relative to the access width. |
|     | <i>width</i> | Access width of value to be read, see <a href="#">Access_width</a> .                           |
| out | <i>value</i> | Return value. If width is smaller than 64 bit, the upper bits are guaranteed to be 0.          |

#### Return values

|                        |                                                                    |
|------------------------|--------------------------------------------------------------------|
| <a href="#">L4_EOK</a> | Success.                                                           |
| -L4_EPERM              | Insufficient read rights.                                          |
| -L4_EINVAL             | Address does not exist or cannot be accessed with the given width. |

### 14.240.3.2 mmio\_write()

```
long L4Re::Mmio_space::mmio_write (
 Addr addr,
 char width,
 l4_uint64_t value)
```

Write a value to the given address.

#### Parameters

|              |                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------|
| <i>addr</i>  | Device virtual address to write to. The address must be aligned relative to the access width. |
| <i>width</i> | Access width of value to write, see <a href="#">Access_width</a> .                            |
| <i>value</i> | Value to write. If width is smaller than 64 bit, the upper bits are ignored.                  |

#### Return values

|                        |                                                                    |
|------------------------|--------------------------------------------------------------------|
| <a href="#">L4_EOK</a> | Success.                                                           |
| -L4_EPERM              | Insufficient write rights.                                         |
| -L4_EINVAL             | Address does not exist or cannot be accessed with the given width. |

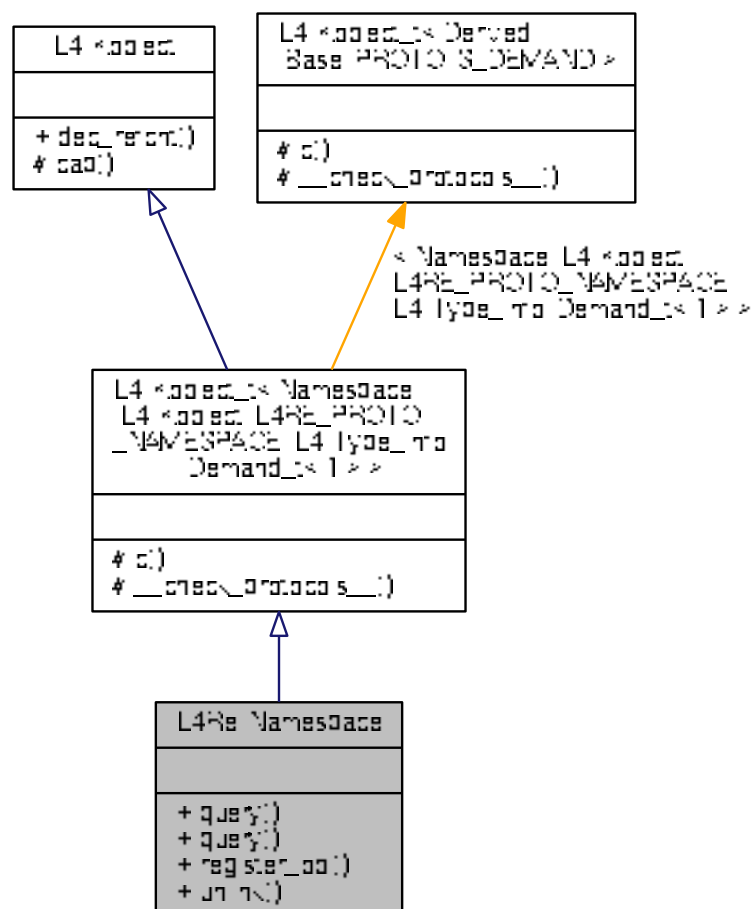
The documentation for this struct was generated from the following file:

- l4/re/mmio\_space

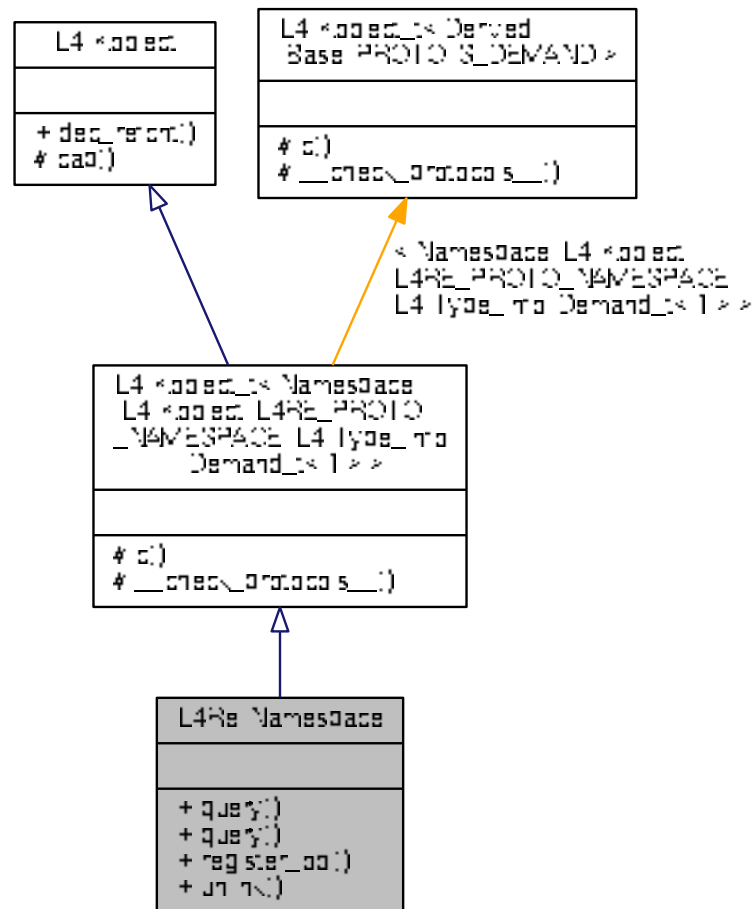
## 14.241 L4Re::Namespace Class Reference

Name-space interface.

Inheritance diagram for L4Re::Namespace:



Collaboration diagram for L4Re::Namespace:



## Public Types

- enum `Register_flags` {  
`Ro` = `L4_CAP_FPAGE_RO`, `Rw` = `L4_CAP_FPAGE_RW`, `Rs` = `L4_CAP_FPAGE_RS`, `Rws` = `L4_CAP_FPAGE_RWS`,  
`Strong` = `L4_CAP_FPAGE_S`, `Trusted` = `0x008`, `Link` = `0x100`, `Overwrite` = `0x200` }  
*Flags for registering name spaces.*
- enum `Query_result_flags` { `Partly_resolved` = `0x020` }  
*Flags returned by query IPC, only used internally.*

## Public Member Functions

- long `query` (char const \*name, `L4::Cap`< void > const &cap, int timeout=To\_default, `l4_umword_t` \*local\_id=0, bool iterate=true) const throw ()  
*Query the name space for a named object.*
- long `query` (char const \*name, unsigned len, `L4::Cap`< void > const &cap, int timeout=To\_default, `l4_umword_t` \*local\_id=0, bool iterate=true) const throw ()



*Query the name space for a named object.*

- long `register_obj` (char const \*name, `L4::lpc::Cap`< void > obj, unsigned flags=`Rw`) const throw ()

*Register an object with a name.*

- long `unlink` (char const \*name)

*Remove an entry from the name space.*

## Additional Inherited Members

### 14.241.1 Detailed Description

Name-space interface.

All name space objects must provide this interface. However, it is not mandatory that a name space object allows to register new capabilities.

The name lookup is done iteratively, this means the hierarchical names are resolved component wise by the client itself.

Definition at line 60 of file `namespace`.

### 14.241.2 Member Enumeration Documentation

#### 14.241.2.1 Query\_result\_flags

```
enum L4Re::Namespace::Query_result_flags
```

Flags returned by query IPC, only used internally.

Enumerator

|                 |                                |
|-----------------|--------------------------------|
| Partly_resolved | Name was only partly resolved. |
|-----------------|--------------------------------|

Definition at line 88 of file `namespace`.

#### 14.241.2.2 Register\_flags

```
enum L4Re::Namespace::Register_flags
```

Flags for registering name spaces.

Enumerator

|    |            |
|----|------------|
| Ro | Read-only. |
|----|------------|

## Enumerator

|           |                                        |
|-----------|----------------------------------------|
| Rw        | Read-write.                            |
| Rs        | Read-only + strong.                    |
| Rws       | Read-write + strong.                   |
| Strong    | Strong.                                |
| Trusted   | Obsolete, do not use.                  |
| Link      | Obsolete, do not use.                  |
| Overwrite | If entry already exists, overwrite it. |

Definition at line 68 of file [namespace](#).

## 14.241.3 Member Function Documentation

14.241.3.1 `query()` [1/2]

```
long L4Re::Namespace::query (
 char const * name,
 L4::Cap< void > const & cap,
 int timeout = To_default,
 l4_umword_t * local_id = 0,
 bool iterate = true) const throw ()
```

Query the name space for a named object.

## Parameters

|     |                 |                                                                                                                                                                                                                  |
|-----|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>name</i>     | String to query (without any leading slashes).                                                                                                                                                                   |
| in  | <i>cap</i>      | Capability slot where the received capability will be put.                                                                                                                                                       |
| in  | <i>timeout</i>  | Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.                                                           |
| out | <i>local_id</i> | If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter. |
| in  | <i>iterate</i>  | If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.                                                                                           |

## Return values

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| 0          | Name could be fully resolved.                                                       |
| >0         | Name could only be partly resolved. The number of remaining characters is returned. |
| -L4_ENOENT | Entry could not be found.                                                           |
| -L4_EAGAIN | Entry exists but no object is yet attached. Try again later.                        |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .                                   |

Definition at line 118 of file [namespace\\_impl.h](#).

## 14.241.3.2 query() [2/2]

```

long L4Re::Namespace::query (
 char const * name,
 unsigned len,
 L4::Cap< void > const & cap,
 int timeout = To_default,
 l4_umword_t * local_id = 0,
 bool iterate = true) const throw ()

```

Query the name space for a named object.

The query string does not necessarily need to be null-terminated.

## Parameters

|     |                 |                                                                                                                                                                                                                  |
|-----|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>len</i>      | Length of the string to query without any terminating null characters.                                                                                                                                           |
| in  | <i>name</i>     | String to query (without any leading slashes).                                                                                                                                                                   |
| in  | <i>cap</i>      | Capability slot where the received capability will be put.                                                                                                                                                       |
| in  | <i>timeout</i>  | Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.                                                           |
| out | <i>local_id</i> | If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter. |
| in  | <i>iterate</i>  | If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.                                                                                           |

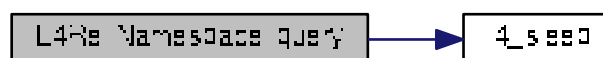
## Return values

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| 0          | Name could be fully resolved.                                                       |
| >0         | Name could only be partly resolved. The number of remaining characters is returned. |
| -L4_ENOENT | Entry could not be found.                                                           |
| -L4_EAGAIN | Entry exists but no object is yet attached. Try again later.                        |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .                                   |

Definition at line 77 of file [namespace\\_impl.h](#).

References [L4\\_EAGAIN](#), [L4\\_EINVAL](#), [l4\\_sleep\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 14.241.3.3 register\_obj()

```
long L4Re::Namespace::register_obj (
 char const * name,
 L4::Ipc::Cap< void > obj,
 unsigned flags = Rw) const throw () [inline]
```

Register an object with a name.

#### Parameters

|              |                                                                                                                                                                                                                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>name</i>  | Name under which the object should be registered.                                                                                                                                                                                                                                                     |
| <i>obj</i>   | Capability to object to register. An invalid capability may be given to only reserve the name for later use.                                                                                                                                                                                          |
| <i>flags</i> | Flags to assign to the entry, see <a href="#">L4Re::Namespace::Register_flags</a> . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space. |

#### Return values

|            |                                                       |
|------------|-------------------------------------------------------|
| 0          | Object was successfully registered with <i>name</i> . |
| -L4_EEXIST | Name already registered.                              |
| -L4_EPERM  | Caller doesn't have necessary permissions.            |
| -L4_ENOMEM | Server has insufficient resources.                    |
| -L4_EINVAL | Invalid parameter.                                    |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .     |

#### Precondition

requires capability rights: {RW}

Definition at line 176 of file [namespace](#).

References [L4\\_RPC\\_NF\\_OP](#).

### 14.241.3.4 unlink()

```
long L4Re::Namespace::unlink (
 char const * name) [inline]
```

Remove an entry from the name space.

#### Parameters

|             |                              |
|-------------|------------------------------|
| <i>name</i> | Name of the entry to remove. |
|-------------|------------------------------|

## Return values

|             |                                                   |
|-------------|---------------------------------------------------|
| 0           | Entry successfully removed.                       |
| -L4_ENOENT  | Given name does not exist.                        |
| -L4_EPERM   | Caller does not have write permission.            |
| -L4_EACCESS | Name cannot be removed.                           |
| <0          | IPC errors, see <a href="#">l4_error_code_t</a> . |

## Precondition

requires capability rights: {RW}

Definition at line 202 of file [namespace](#).

The documentation for this class was generated from the following files:

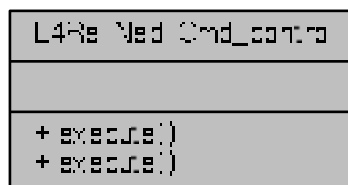
- [l4/re/namespace](#)
- [l4/re/impl/namespace\\_impl.h](#)

## 14.242 L4Re::Ned::Cmd\_control Class Reference

Direct control interface for Ned.

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

Collaboration diagram for L4Re::Ned::Cmd\_control:



### Public Member Functions

- long [execute](#) (L4::lpc::String<> cmd) noexcept  
*Execute the given Lua code.*
- long [execute](#) (L4::lpc::String<> cmd, L4::lpc::String< char > \*result) noexcept  
*Execute the given Lua code.*

### 14.242.1 Detailed Description

Direct control interface for Ned.

Definition at line 20 of file [cmd\\_control](#).

### 14.242.2 Member Function Documentation

#### 14.242.2.1 `execute()` [1/2]

```
long L4Re::Ned::Cmd_control::execute (
 L4::Ipc::String<> cmd) [inline], [noexcept]
```

Execute the given Lua code.

##### Parameters

|    |            |                                  |
|----|------------|----------------------------------|
| in | <i>cmd</i> | String with Lua code to execute. |
|----|------------|----------------------------------|

##### Return values

|                   |                                 |
|-------------------|---------------------------------|
| <i>L4_EOK</i>     | Code was successfully executed. |
| <i>-L4_EINVAL</i> | Code could not be parsed.       |
| <i>-L4_EIO</i>    | Error during code execution.    |

The code is executed using the global Lua state of ned which is retained between successive calls to `execute`. Thus you may define data in one call to `execute` and use it in a subsequent call.

This function does not return any results from the execution of the Lua code itself.

Definition at line 43 of file [cmd\\_control](#).

#### 14.242.2.2 `execute()` [2/2]

```
long L4Re::Ned::Cmd_control::execute (
 L4::Ipc::String<> cmd,
 L4::Ipc::String< char > * result) [inline], [noexcept]
```

Execute the given Lua code.

##### Parameters

|     |               |                                                         |
|-----|---------------|---------------------------------------------------------|
| in  | <i>cmd</i>    | String with Lua code to execute.                        |
| out | <i>result</i> | The first return value of the Lua code block as string. |

## Return values

|                         |                                 |
|-------------------------|---------------------------------|
| <code>L4_EOK</code>     | Code was successfully executed. |
| <code>-L4_EINVAL</code> | Code could not be parsed.       |
| <code>-L4_EIO</code>    | Error during code execution.    |

The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

Definition at line 65 of file [cmd\\_control](#).

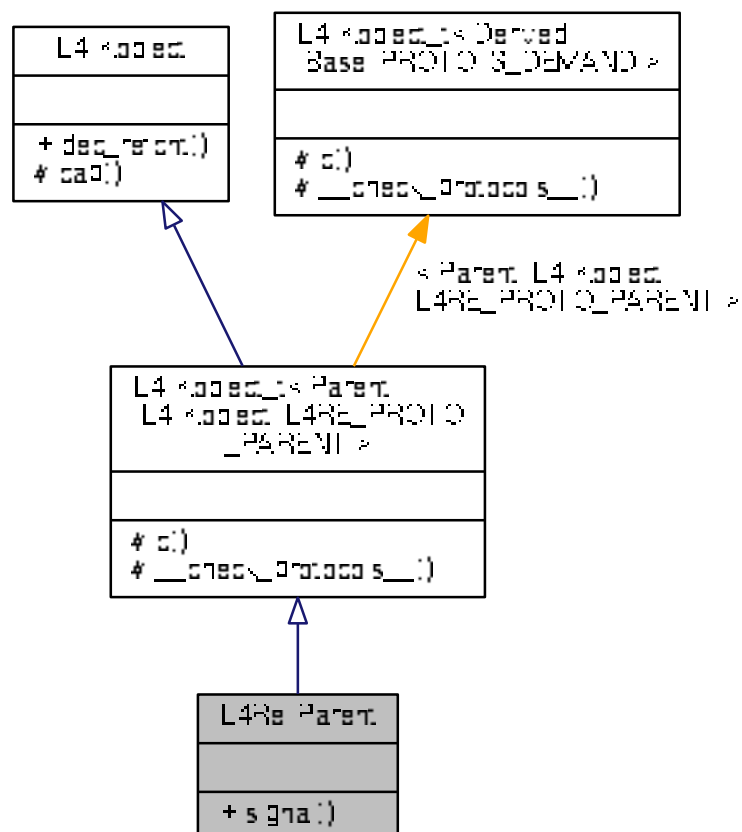
The documentation for this class was generated from the following file:

- `pkg/l4re-core/ned/lib/include/cmd_control`

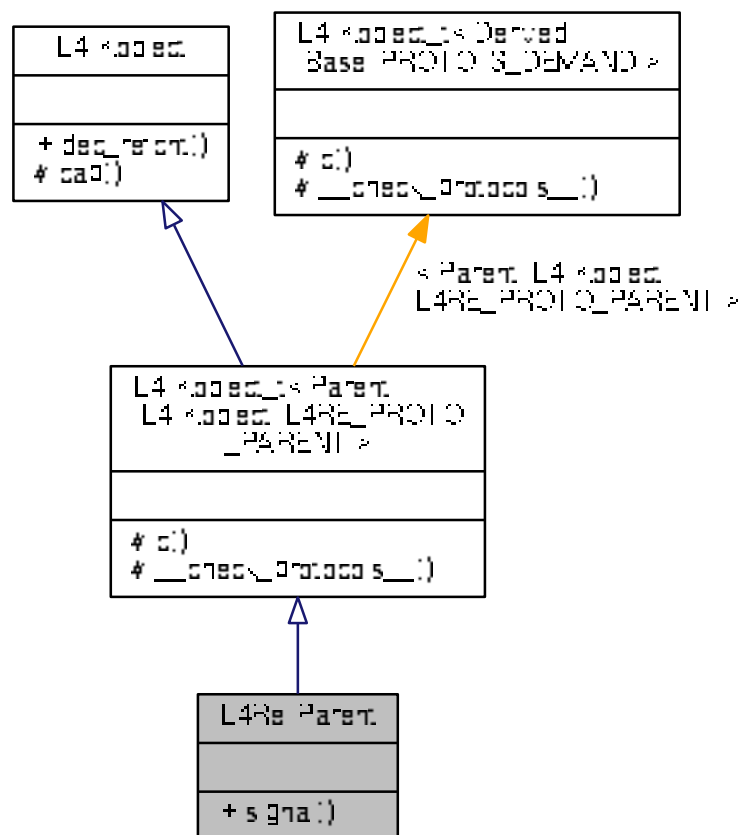
## 14.243 L4Re::Parent Class Reference

[Parent](#) interface.

Inheritance diagram for L4Re::Parent:



Collaboration diagram for L4Re::Parent:



## Public Member Functions

- long [signal](#) (unsigned long sig, unsigned long val)  
*Send a signal to the parent.*

## Additional Inherited Members

### 14.243.1 Detailed Description

[Parent](#) interface.

See also

[Parent API](#) for more details about the purpose.

Definition at line 51 of file [parent](#).



## 14.243.2 Member Function Documentation

### 14.243.2.1 signal()

```
long L4Re::Parent::signal (
 unsigned long sig,
 unsigned long val)
```

Send a signal to the parent.

#### Parameters

|            |                     |
|------------|---------------------|
| <i>sig</i> | Signal to send      |
| <i>val</i> | Value of the signal |

#### Returns

0 on success, <0 on error

- [-L4\\_ENOREPLY](#)
- IPC errors

The documentation for this class was generated from the following file:

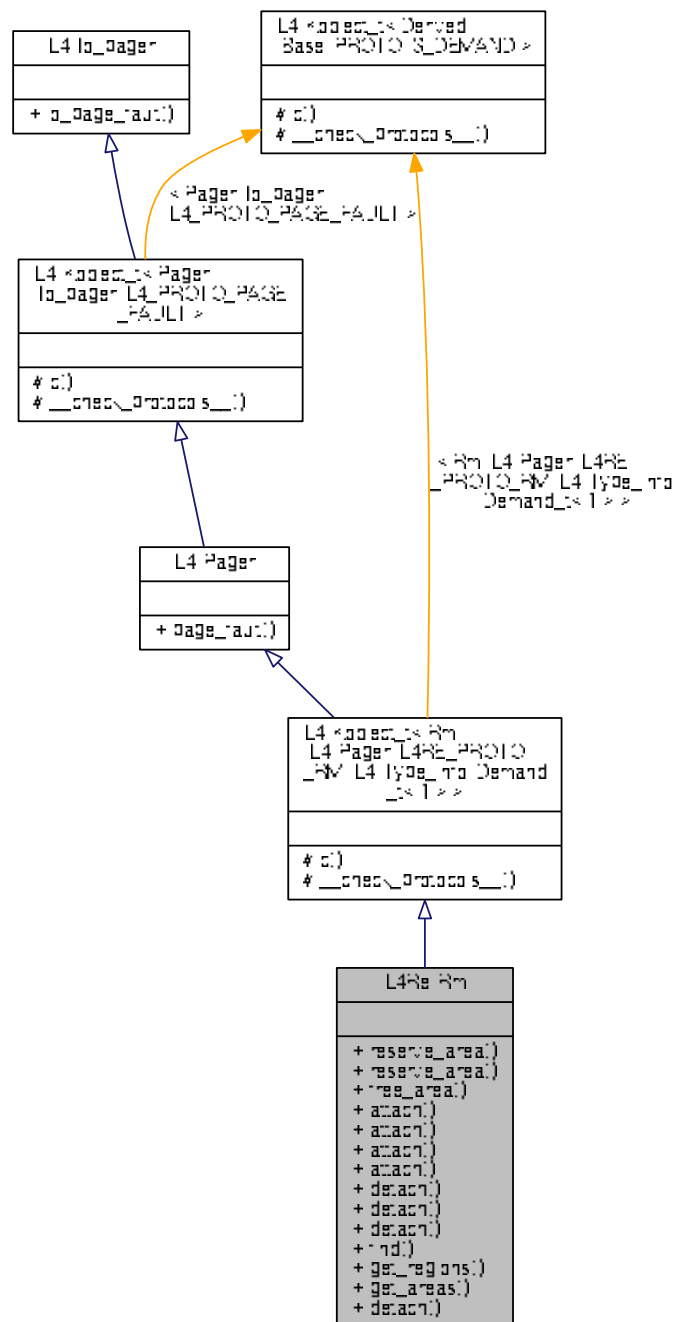
- [l4/re/parent](#)

## 14.244 L4Re::Rm Class Reference

Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:





- Generated for L4Re by Doxygen

```
Dataspace::Map_caching_mask << Caching_ds_shift, Cache_normal = Dataspace::Map_normal <<
Caching_ds_shift,
Cache_buffered = Dataspace::Map_bufferable << Caching_ds_shift, Cache_uncached = Dataspace::
Map_uncacheable << Caching_ds_shift, Region_flags = Caching | 0x0f }
```

*Flags for regions.*

- enum [Attach\\_flags](#) { [Search\\_addr](#) = 0x20, [In\\_area](#) = 0x40, [Eager\\_map](#) = 0x80, [Attach\\_flags](#) = 0xf0 }

*Flags for attach operation.*

- enum [Detach\\_flags](#) { [Detach\\_exact](#) = 1, [Detach\\_overlap](#) = 2, [Detach\\_keep](#) = 4 }

*Flags for detach operation.*

## Public Member Functions

- long [reserve\\_area](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, unsigned flags=0, unsigned char align=[L4\\_PAGE←SHIFT](#)) const throw ()

*Reserve the given area in the region map.*

- template<typename T >  
long [reserve\\_area](#) (T \*\*start, unsigned long size, unsigned flags=0, unsigned char align=[L4\\_PAGESHIFT](#)) const throw ()

*Reserve the given area in the region map.*

- long [free\\_area](#) ([l4\\_addr\\_t](#) addr)

*Free an area from the region map.*

- long [attach](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, unsigned long flags, [L4::lpc::Cap](#)< [Dataspace](#) > mem, [l4\\_addr\\_t](#) offs=0, unsigned char align=[L4\\_PAGESHIFT](#)) const throw ()

*Attach a data space to a region.*

- template<typename T >  
long [attach](#) (T \*\*start, unsigned long size, unsigned long flags, [L4::lpc::Cap](#)< [Dataspace](#) > mem, [l4\\_addr\\_t](#) offs=0, unsigned char align=[L4\\_PAGESHIFT](#)) const throw ()

*Attach a data space to a region.*

- int [detach](#) ([l4\\_addr\\_t](#) addr, [L4::Cap](#)< [Dataspace](#) > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task=This\_task) const throw ()

*Detach a region from the address space.*

- int [detach](#) (void \*addr, [L4::Cap](#)< [Dataspace](#) > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task=This\_task) const throw ()

*Detach a region from the address space.*

- int [detach](#) ([l4\\_addr\\_t](#) start, unsigned long size, [L4::Cap](#)< [Dataspace](#) > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task) const throw ()

*Detach all regions of the specified interval.*

- int [find](#) ([l4\\_addr\\_t](#) \*addr, unsigned long \*size, [l4\\_addr\\_t](#) \*offset, unsigned \*flags, [L4::Cap](#)< [Dataspace](#) > \*m) throw ()

*Find a region given an address and size.*

## Additional Inherited Members

### 14.244.1 Detailed Description

Region map.

See also

[Region map API](#) .

Definition at line 71 of file [rm](#).

## 14.244.2 Member Enumeration Documentation

### 14.244.2.1 Attach\_flags

enum [L4Re::Rm::Attach\\_flags](#)

Flags for attach operation.

#### Enumerator

|              |                                         |
|--------------|-----------------------------------------|
| Search_addr  | Search for a suitable address range.    |
| In_area      | Search only in area, or map into area.  |
| Eager_map    | Eagerly map the attached data space in. |
| Attach_flags | Mask of all attach flags.               |

Definition at line [113](#) of file [rm](#).

### 14.244.2.2 Detach\_flags

enum [L4Re::Rm::Detach\\_flags](#)

Flags for detach operation.

#### Enumerator

|                |                                                                               |
|----------------|-------------------------------------------------------------------------------|
| Detach_exact   | Do an unmap of the exact region given.                                        |
| Detach_overlap | Do an unmap of all overlapping regions.                                       |
| Detach_keep    | Do not free the detached data space, ignore the <a href="#">Detach_free</a> . |

Definition at line [123](#) of file [rm](#).

### 14.244.2.3 Detach\_result

enum [L4Re::Rm::Detach\\_result](#)

Result values for detach operation.

#### Enumerator

|              |                                  |
|--------------|----------------------------------|
| Detached_ds  | Detached data sapce.             |
| Kept_ds      | Kept data space.                 |
| Split_ds     | Splitted data space, and done.   |
| Detach_again | Detached data space, more to do. |

Definition at line 77 of file [rm](#).

#### 14.244.2.4 Region\_flags

```
enum L4Re::Rm::Region_flags
```

Flags for regions.

##### Enumerator

|                  |                                                                             |
|------------------|-----------------------------------------------------------------------------|
| Read_only        | Region is read-only.                                                        |
| Detach_free      | Free the portion of the data space after detach.                            |
| Pager            | Region has a pager.                                                         |
| Reserved         | Region is reserved (blocked)                                                |
| Caching_shift    | Start of <a href="#">Rm</a> cache bits.                                     |
| Caching_ds_shift | Shift value for <a href="#">Dataspace</a> to <a href="#">Rm</a> cache bits. |
| Caching          | Mask of all <a href="#">Rm</a> cache bits.                                  |
| Cache_normal     | Cache bits for normal cacheable memory.                                     |
| Cache_buffered   | Cache bits for buffered (write combining) memory.                           |
| Cache_uncached   | Cache bits for uncached memory.                                             |
| Region_flags     | Mask of all region flags.                                                   |

Definition at line 88 of file [rm](#).

### 14.244.3 Member Function Documentation

#### 14.244.3.1 attach() [1/2]

```
long L4Re::Rm::attach (
 l4_addr_t * start,
 unsigned long size,
 unsigned long flags,
 L4::Ipc::Cap< Dataspace > mem,
 l4_addr_t offs = 0,
 unsigned char align = L4_PAGESHIFT) const throw ()
```

Attach a data space to a region.

##### Parameters

|         |              |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>start</i> | Virtual start address where the region manager shall attach the data space. If <a href="#">L4Re::Rm::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to. |
|---------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Parameters

|  |              |                                                                                                                                                                                                                                               |
|--|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <i>size</i>  | Size of the data space to attach (in bytes)                                                                                                                                                                                                   |
|  | <i>flags</i> | Flags, see <a href="#">L4Re::Rm::Attach_flags</a> and <a href="#">L4Re::Rm::Region_flags</a> . If the <code>Eager_map</code> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails. |
|  | <i>mem</i>   | Data space                                                                                                                                                                                                                                    |
|  | <i>offs</i>  | Offset into the data space to use                                                                                                                                                                                                             |
|  | <i>align</i> | Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::Search_addr</a> flag is used.                                                              |

## Return values

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| 0                 | Success                                                         |
| -L4_ENOENT        | No area could be found (see <a href="#">L4Re::Rm::In_area</a> ) |
| -L4_EPERM         | Operation not allowed.                                          |
| -L4_EINVAL        |                                                                 |
| -L4_EADDRNOTAVAIL | The given address is not available.                             |
| <0                | IPC errors                                                      |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

## Note

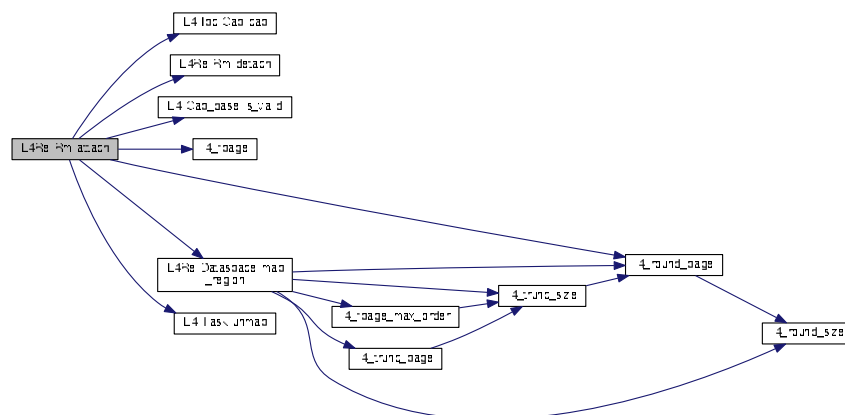
When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 43 of file [rm\\_impl.h](#).

References [L4::ipc::Cap< T >::cap\(\)](#), [detach\(\)](#), [L4::Cap\\_base::is\\_valid\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), [I4\\_fpage\(\)](#), [L4::\\_FPAGE\\_RWX](#), [L4\\_INVALID\\_CAP](#), [L4\\_LOG2\\_PAGESIZE](#), [I4\\_round\\_page\(\)](#), [L4\\_UNLIKELY](#), [L4Re::Dataspace::map\\_region\(\)](#), [L4Re::Dataspace::Map\\_ro](#), [L4Re::Dataspace::Map\\_rw](#), and [L4::Task::unmap\(\)](#).

Here is the call graph for this function:



14.244.3.2 `attach()` [2/2]

```

template<typename T >
long L4Re::Rm::attach (
 T ** start,
 unsigned long size,
 unsigned long flags,
 L4::Ipc::Cap< Dataspace > mem,
 l4_addr_t offs = 0,
 unsigned char align = L4_PAGESHIFT) const throw () [inline]

```

Attach a data space to a region.

## Parameters

|                |              |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>in, out</i> | <i>start</i> | Virtual start address where the region manager shall attach the data space. If <a href="#">L4Re::Rm::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to. |
|                | <i>size</i>  | Size of the data space to attach (in bytes)                                                                                                                                                                                                                                                                                                                                                                                        |
|                | <i>flags</i> | Flags, see <a href="#">L4Re::Rm::Attach_flags</a> and <a href="#">L4Re::Rm::Region_flags</a> . If the <code>Eager_map</code> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.                                                                                                                                                                                      |
|                | <i>mem</i>   | Data space                                                                                                                                                                                                                                                                                                                                                                                                                         |
|                | <i>offs</i>  | Offset into the data space to use                                                                                                                                                                                                                                                                                                                                                                                                  |
|                | <i>align</i> | Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::Search_addr</a> flag is used.                                                                                                                                                                                                                                                   |

## Return values

|                          |                                                                 |
|--------------------------|-----------------------------------------------------------------|
| <i>0</i>                 | Success                                                         |
| <i>-L4_ENOENT</i>        | No area could be found (see <a href="#">L4Re::Rm::In_area</a> ) |
| <i>-L4_EPERM</i>         | Operation not allowed.                                          |
| <i>-L4_EINVAL</i>        |                                                                 |
| <i>-L4_EADDRNOTAVAIL</i> | The given address is not available.                             |
| <i>&lt;0</i>             | IPC errors                                                      |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

## Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

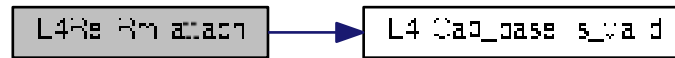
There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 347 of file [rm](#).

References [L4::Cap\\_base::is\\_valid\(\)](#), and [L4\\_PAGESHIFT](#).



Here is the call graph for this function:



### 14.244.3.3 detach() [1/3]

```

int L4Re::Rm::detach (
 l4_addr_t addr,
 L4::Cap< Dataspace > * mem,
 L4::Cap< L4::Task > const & task = This_task) const throw () [inline]

```

Detach a region from the address space.

#### Parameters

|     |             |                                                                                                                                                                    |
|-----|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>addr</i> | Virtual address of region, any address within the region is valid.                                                                                                 |
| out | <i>mem</i>  | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

#### Return values

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <code>-L4_ENOENT</code>       | No region found. |
| <code>&lt;0</code>            | IPC errors       |

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 591 of file [rm](#).

Referenced by [attach\(\)](#).

Here is the caller graph for this function:



## 14.244.3.4 detach() [2/3]

```
int L4Re::Rm::detach (
 void * addr,
 L4::Cap< Dataspace > * mem,
 L4::Cap< L4::Task > const & task = This_task) const throw () [inline]
```

Detach a region from the address space.

## Parameters

|     |             |                                                                                                                                                                    |
|-----|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>addr</i> | Virtual address of region, any address within the region is valid.                                                                                                 |
| out | <i>mem</i>  | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

## Return values

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <a href="#">-L4_ENOENT</a>    | No region found. |
| <a href="#">&lt;0</a>         | IPC errors       |

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 596 of file [rm](#).

## 14.244.3.5 detach() [3/3]

```
int L4Re::Rm::detach (
 l4_addr_t start,
 unsigned long size,
 L4::Cap< Dataspace > * mem,
 L4::Cap< L4::Task > const & task) const throw () [inline]
```

Detach all regions of the specified interval.

## Parameters

|     |              |                                                                                                                                                                    |
|-----|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>start</i> | Start of area to detach, must be within region.                                                                                                                    |
|     | <i>size</i>  | Size of of area to detach (in bytes).                                                                                                                              |
| out | <i>mem</i>   | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i>  | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

## Return values

|                                      |                  |
|--------------------------------------|------------------|
| <a href="#"><i>Detach_result</i></a> | On success.      |
| <a href="#"><i>-L4_ENOENT</i></a>    | No region found. |
| $<0$                                 | IPC errors       |

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line 601 of file [rm](#).

## 14.244.3.6 find()

```
int L4Re::Rm::find (
 l4_addr_t * addr,
 unsigned long * size,
 l4_addr_t * offset,
 unsigned * flags,
 L4::Cap< Dataspace > * m) throw () [inline]
```

Find a region given an address and size.

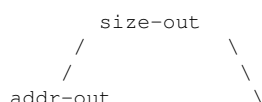
## Parameters

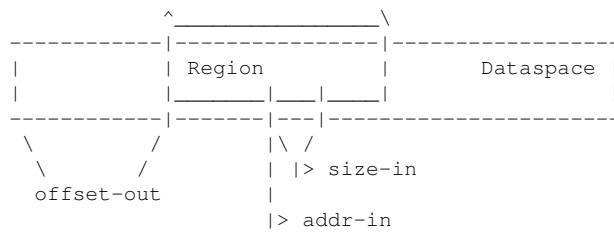
|     |               |                                                                                |
|-----|---------------|--------------------------------------------------------------------------------|
|     | <i>addr</i>   | Address to look for                                                            |
|     | <i>size</i>   | Size of the area to look for (in bytes).                                       |
| out | <i>addr</i>   | Start address of the found region.                                             |
| out | <i>size</i>   | Size of the found region (in bytes).                                           |
| out | <i>offset</i> | Offset at the beginning of the region within the associated dataspace.         |
| out | <i>flags</i>  | Region flags, see <a href="#">Region_flags</a> (and <a href="#">ln_area</a> ). |
| out | <i>m</i>      | Associated dataspace or paging service.                                        |

## Return values

|                                   |                        |
|-----------------------------------|------------------------|
| $0$                               | Success                |
| <a href="#"><i>-L4_EPERM</i></a>  | Operation not allowed. |
| <a href="#"><i>-L4_ENOENT</i></a> | No region found.       |
| $<0$                              | IPC errors             |

This function returns the properties of the region that contains the area described by the *addr* and *size* parameter. If no such region is found but a reserved area, the area is returned and [ln\\_area](#) is set in 'flags'. Note, in the case of an area the 'offset' and 'm' return values are invalid.



**Note**

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 555 of file [rm](#).

References [L4\\_RPC](#), and [L4\\_RPC\\_NF](#).

**14.244.3.7 free\_area()**

```
long L4Re::Rm::free_area (
 l4_addr_t addr)
```

Free an area from the region map.

**Parameters**

|             |                                     |
|-------------|-------------------------------------|
| <i>addr</i> | An address within the area to free. |
|-------------|-------------------------------------|

**Return values**

|            |                |
|------------|----------------|
| 0          | Success        |
| -L4_ENOENT | No area found. |
| <0         | IPC errors     |

**Note**

The data spaces that are attached to that area are not detached by this operation.

**See also**

[reserve\\_area\(\)](#) for more information about areas.

**14.244.3.8 reserve\_area()** [1/2]

```
long L4Re::Rm::reserve_area (
 l4_addr_t * start,
```

```

unsigned long size,
unsigned flags = 0,
unsigned char align = L4_PAGESHIFT) const throw) [inline]

```

Reserve the given area in the region map.

#### Parameters

|                |              |                                                                                                                       |
|----------------|--------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>in, out</i> | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area.                              |
|                | <i>size</i>  | The size of the area to reserve (in bytes).                                                                           |
|                | <i>flags</i> | Flags for the reserved area (see <a href="#">L4Re::Rm::Region_flags</a> and <a href="#">L4Re::Rm::Attach_flags</a> ). |
|                | <i>align</i> | Alignment of area if searched as bits (log2 value).                                                                   |

#### Return values

|                          |                                    |
|--------------------------|------------------------------------|
| <i>0</i>                 | Success                            |
| <i>-L4_EADDRNOTAVAIL</i> | The given area cannot be reserved. |
| <i>&lt;0</i>             | IPC errors                         |

This function reserves an area within the virtual address space implemented by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [ln\\_area](#) flag and a start address within the area itself.

#### Note

When searching for a free place in the virtual address space (with *flags* = *Search\_addr*), the space between *start* and the end of the virtual address space is searched.

Definition at line [241](#) of file [rm](#).

References [L4\\_RPC\\_NF](#).

#### 14.244.3.9 reserve\_area() [2/2]

```

template<typename T >
long L4Re::Rm::reserve_area (
 T ** start,
 unsigned long size,
 unsigned flags = 0,
 unsigned char align = L4_PAGESHIFT) const throw) [inline]

```

Reserve the given area in the region map.

## Parameters

|                |              |                                                                                                   |
|----------------|--------------|---------------------------------------------------------------------------------------------------|
| <i>in, out</i> | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area.          |
|                | <i>size</i>  | The size of the area to reserve (in bytes).                                                       |
|                | <i>flags</i> | Flags for the reserved area (see <a href="#">Region_flags</a> and <a href="#">Attach_flags</a> ). |
|                | <i>align</i> | Alignment of area if searched as bits (log2 value).                                               |

## Return values

|                          |                                    |
|--------------------------|------------------------------------|
| <i>0</i>                 | Success                            |
| <i>-L4_EADDRNOTAVAIL</i> | The given area cannot be reserved. |
| <i>&lt;0</i>             | IPC errors                         |

For more information, please refer to the analogous function

## See also

[L4Re::Rm::reserve\\_area](#).

Definition at line 267 of file [rm](#).

References [L4\\_PAGESHIFT](#), [L4\\_RPC](#), and [L4\\_RPC\\_NF](#).

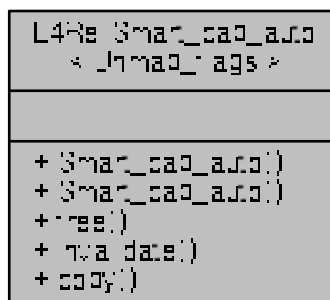
The documentation for this class was generated from the following files:

- [l4/re/rm](#)
- [l4/re/impl/rm\\_impl.h](#)

## 14.245 L4Re::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for Auto\_cap and Auto\_del\_cap.

Collaboration diagram for L4Re::Smart\_cap\_auto< Unmap\_flags >:



### 14.245.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_cap_auto< Unmap_flags >
```

Helper for Auto\_cap and Auto\_del\_cap.

Definition at line 147 of file [cap\\_alloc](#).

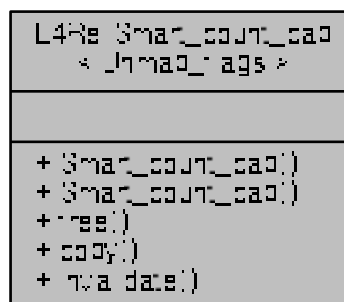
The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

## 14.246 L4Re::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for Ref\_cap and Ref\_del\_cap.

Collaboration diagram for L4Re::Smart\_count\_cap< Unmap\_flags >:



### Public Member Functions

- `void free (L4::Cap_base &c) throw ()`  
Free operation for [L4::Smart\\_cap](#) (decrement ref count and delete if 0).
- `L4::Cap_base copy (L4::Cap_base const &src)`  
Copy operation for [L4::Smart\\_cap](#) (increment ref count).

### Static Public Member Functions

- `static void invalidate (L4::Cap_base &c) throw ()`  
Invalidate operation for [L4::Smart\\_cap](#).

### 14.246.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_count_cap< Unmap_flags >
```

Helper for Ref\_cap and Ref\_del\_cap.

Definition at line 182 of file [cap\\_alloc](#).

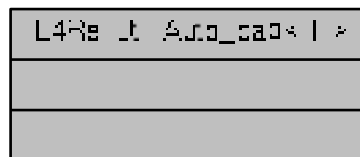
The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

### 14.247 L4Re::Util::Auto\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

Collaboration diagram for L4Re::Util::Auto\_cap< T >:



### 14.247.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Auto_cap< T >
```

Automatic capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

|          |                                                        |
|----------|--------------------------------------------------------|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--------------------------------------------------------|

**Deprecated** Use [L4Re::Util::Unique\\_cap](#).

This kind of automatic capability is useful for capabilities that shall have a lifetime that is strictly coupled to one C++ scope.

Usage:



```

{
 L4Re::Util::Auto_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());

 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

 ...

 // At the end of the scope ds_cap is unmapped and the capability selector
 // is freed.
}

```

Definition at line 163 of file [cap\\_alloc](#).

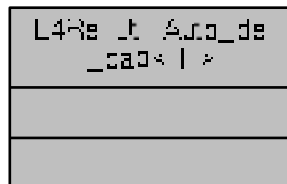
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 14.248 L4Re::Util::Auto\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Collaboration diagram for L4Re::Util::Auto\_del\_cap< T >:



### 14.248.1 Detailed Description

```

template<typename T>
struct L4Re::Util::Auto_del_cap< T >

```

Automatic capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

|          |                                                        |
|----------|--------------------------------------------------------|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--------------------------------------------------------|

**Deprecated** Use [L4Re::Util::Unique\\_cap](#).

This kind of automatic capability is useful for capabilities with that shall have a lifetime that is strictly coupled to one C++ scope. The main difference to [Auto\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
 L4Re::Util::Auto_del_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());

 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

 ...

 // At the end of the scope ds_cap is unmapped and the capability selector
 // is freed. Because the deletion flag is set the data space shall be
 // also deleted (even if there are other references to this data space).
}
```

Definition at line 200 of file [cap\\_alloc](#).

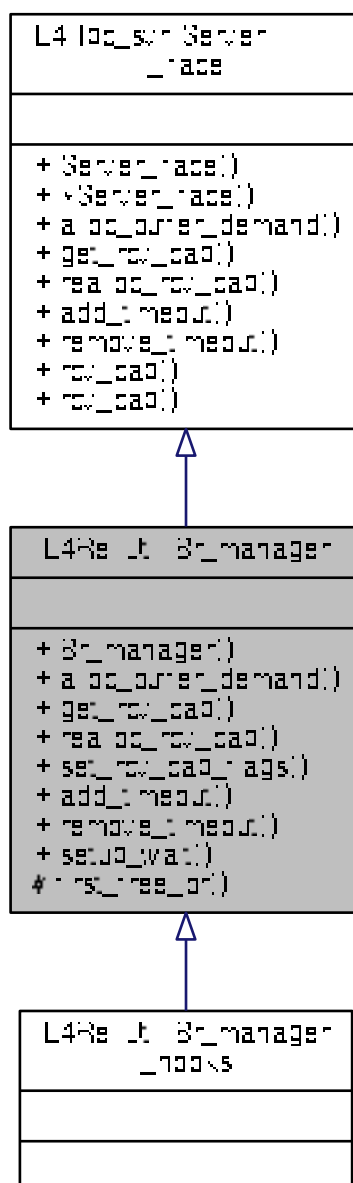
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

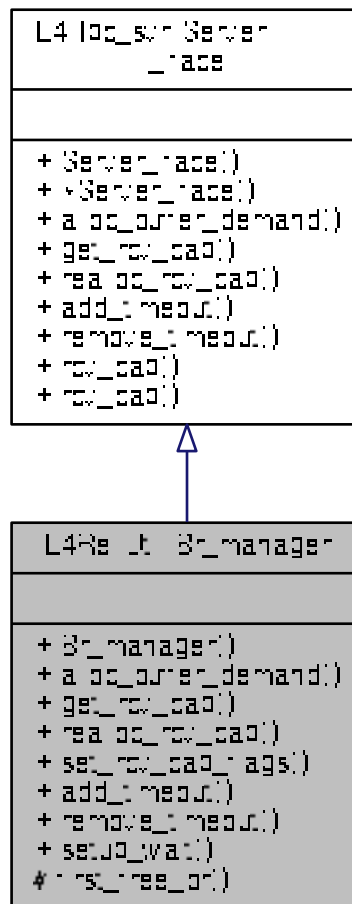
## 14.249 L4Re::Util::Br\_manager Class Reference

Buffer-register (BR) manager for [L4::Server](#).

Inheritance diagram for L4Re::Util::Br\_manager:



Collaboration diagram for L4Re::Util::Br\_manager:



## Public Member Functions

- [Br\\_manager](#) ()  
*Make a buffer-register (BR) manager.*
- `int` [alloc\\_buffer\\_demand](#) ([Demand](#) const &d)  
*Tells the server to allocate buffers for the given demand.*
- `L4::Cap< void >` [get\\_rcv\\_cap](#) (int i) const  
*Get capability slot allocated to the given receive buffer.*
- `int` [realloc\\_rcv\\_cap](#) (int i)  
*Allocate a new capability for the given receive buffer.*
- `void` [set\\_rcv\\_cap\\_flags](#) (unsigned long flags)  
*Set the receive flags for the buffers.*
- `int` [add\\_timeout](#) (`L4::lpc_svr::Timeout *`, `l4_kernel_clock_t`)  
*No timeouts handled by us.*
- `int` [remove\\_timeout](#) (`L4::lpc_svr::Timeout *`)  
*No timeouts handled by us.*
- `void` [setup\\_wait](#) (`l4_utcb_t *utcb`, `L4::lpc_svr::Reply_mode`)  
*setup\_wait() used the server loop (L4::Server)*

## Protected Member Functions

- unsigned [first\\_free\\_br](#) () const  
*Used for assigning BRs for a timeout.*

## Additional Inherited Members

### 14.249.1 Detailed Description

Buffer-register (BR) manager for [L4::Server](#).

Implementation of the [L4::lpc\\_svr::Server\\_iface](#) API for managing the server-side receive buffers needed for a set of server objects running within a server.

Definition at line 36 of file [br\\_manager](#).

### 14.249.2 Member Function Documentation

#### 14.249.2.1 [alloc\\_buffer\\_demand\(\)](#)

```
int L4Re::Util::Br_manager::alloc_buffer_demand (
 Demand const & demand) [inline], [virtual]
```

Tells the server to allocate buffers for the given demand.

#### Parameters

|               |                                                                                           |
|---------------|-------------------------------------------------------------------------------------------|
| <i>demand</i> | The total server-side demand of receive buffers needed for a given interface, see Demand. |
|---------------|-------------------------------------------------------------------------------------------|

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 50 of file [br\\_manager](#).

#### 14.249.2.2 [get\\_rcv\\_cap\(\)](#)

```
L4::Cap<void> L4Re::Util::Br_manager::get_rcv_cap (
 int index) const [inline], [virtual]
```

Get capability slot allocated to the given receive buffer.

**Parameters**

|              |                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>index</i> | The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ). |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

**Returns**

Capability slot currently allocated to the given receive buffer.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 81 of file [br\\_manager](#).

References [L4\\_CAP\\_MASK](#).

**14.249.2.3 realloc\_rcv\_cap()**

```
int L4Re::Util::Br_manager::realloc_rcv_cap (
 int index) [inline], [virtual]
```

Allocate a new capability for the given receive buffer.

**Parameters**

|              |                                                                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>index</i> | The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ). |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

**Returns**

0 on success, < 0 on error.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 89 of file [br\\_manager](#).

## 14.249.2.4 set\_rcv\_cap\_flags()

```
void L4Re::Util::Br_manager::set_rcv_cap_flags (
 unsigned long flags) [inline]
```

Set the receive flags for the buffers.

## Precondition

Must be called before any handlers are registered.

## Parameters

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| <i>flags</i> | New receive capability flags, see <a href="#">l4_msg_item_consts_t</a> . |
|--------------|--------------------------------------------------------------------------|

Definition at line 113 of file [br\\_manager](#).

References [l4\\_assert](#).

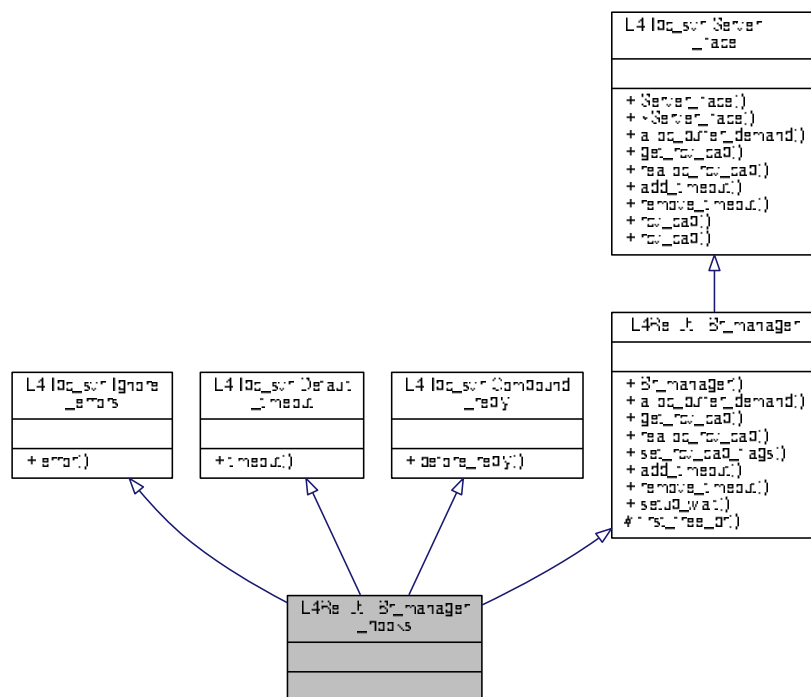
The documentation for this class was generated from the following file:

- l4/re/util/br\_manager

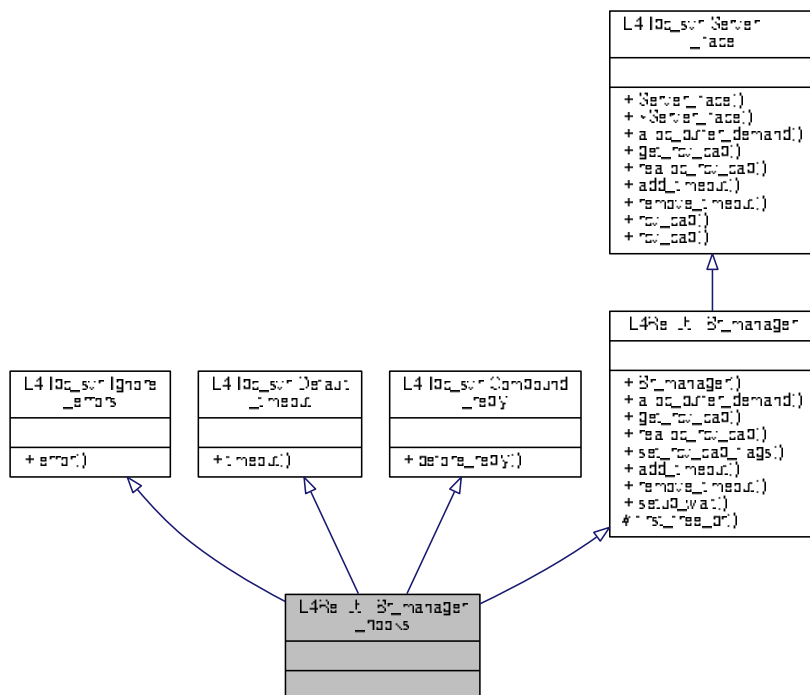
## 14.250 L4Re::Util::Br\_manager\_hooks Struct Reference

Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

Inheritance diagram for L4Re::Util::Br\_manager\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_hooks:



## Additional Inherited Members

### 14.250.1 Detailed Description

Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

This class can be used whenever a server loop including full management of receive buffer resources is needed.

Definition at line 160 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

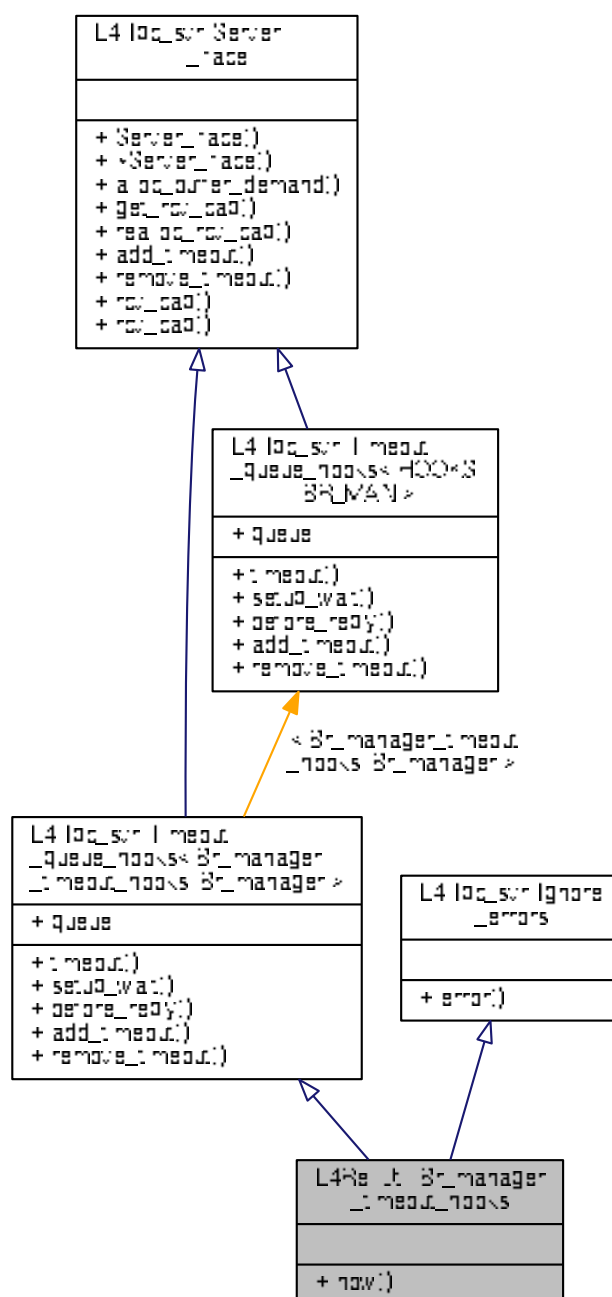
- [l4/re/util/br\\_manager](#)

### 14.251 L4Re::Util::Br\_manager\_timeout\_hooks Struct Reference

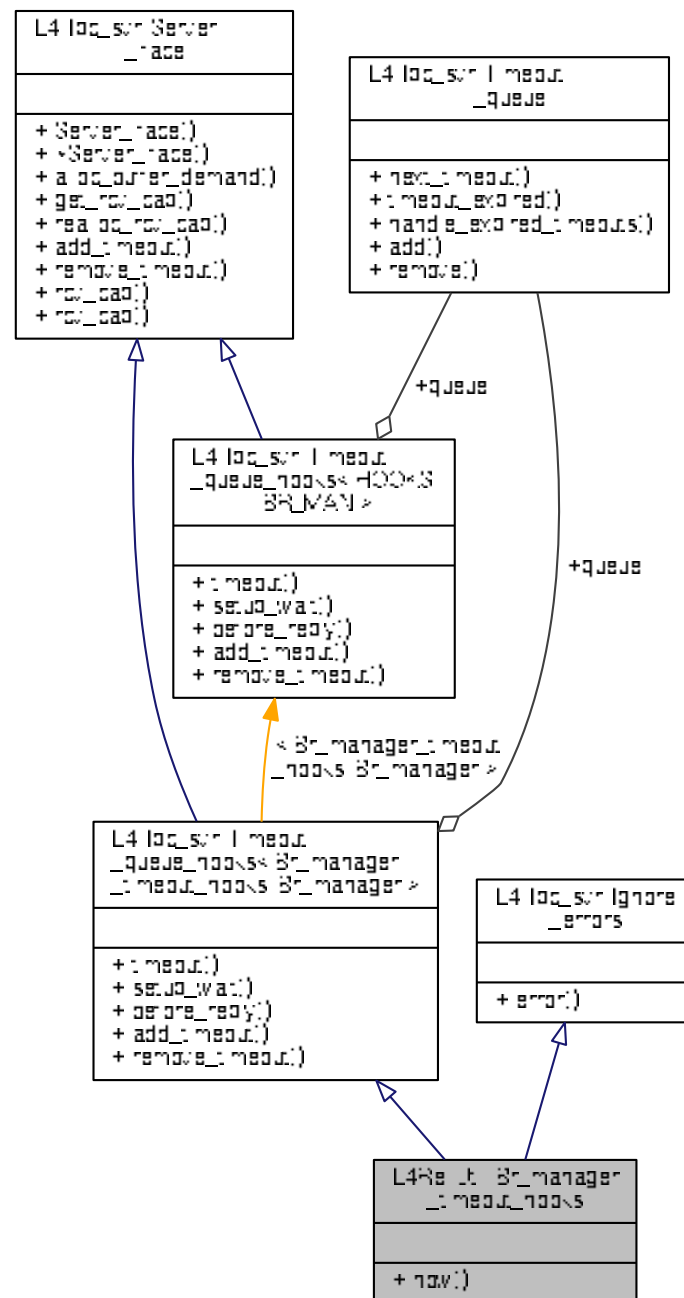
Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.



Inheritance diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



## Additional Inherited Members

### 14.251.1 Detailed Description

Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.

This class can be used for server loops that need the full package of buffer-register management and a timeout queue.

Definition at line 174 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

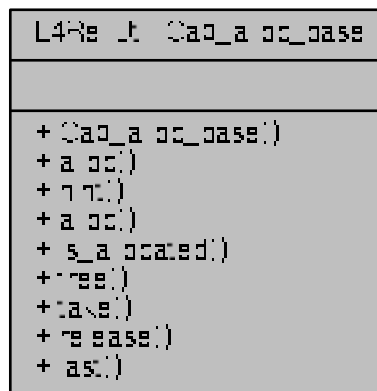
- [l4/re/util/br\\_manager](#)

## 14.252 L4Re::Util::Cap\_alloc\_base Class Reference

Capability allocator.

Inherited by L4Re::Util::Cap\_alloc< Size >.

Collaboration diagram for L4Re::Util::Cap\_alloc\_base:



### Public Member Functions

- `template<typename T >`  
[L4::Cap< T > alloc](#) () throw ()  
*Allocate a capability slot.*
- `template<typename T >`  
`void free` (L4::Cap< T > const &cap, [l4\\_cap\\_idx\\_t](#) task=[L4\\_INVALID\\_CAP](#), [l4\\_umword\\_t](#) unmap\_flags=[L4\\_↵](#)  
[\\_FP\\_ALL\\_SPACES](#)) throw ()  
*Free a capability slot.*

### 14.252.1 Detailed Description

Capability allocator.

Definition at line 38 of file [bitmap\\_cap\\_alloc](#).

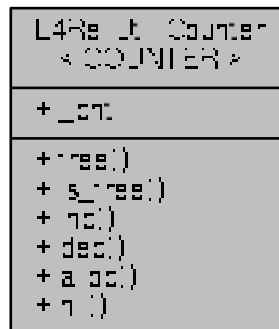
The documentation for this class was generated from the following file:

- [l4/re/util/bitmap\\_cap\\_alloc](#)

## 14.253 L4Re::Util::Counter< COUNTER > Struct Template Reference

[Counter](#) for [Counting\\_cap\\_alloc](#) with variable data width.

Collaboration diagram for L4Re::Util::Counter< COUNTER >:



### 14.253.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter< COUNTER >
```

[Counter](#) for [Counting\\_cap\\_alloc](#) with variable data width.

Definition at line 36 of file [counting\\_cap\\_alloc](#).

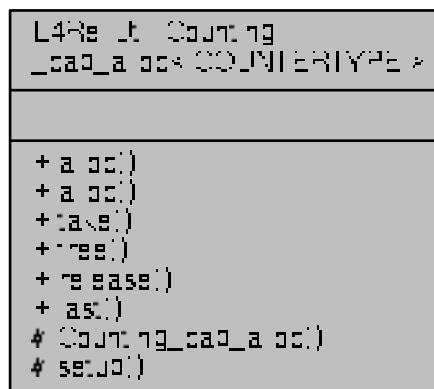
The documentation for this struct was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

## 14.254 L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE > Class Template Reference

Internal reference-counting cap allocator.

Collaboration diagram for L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE >:



## Public Member Functions

- [L4::Cap< void > alloc \(\)](#) throw ()  
*Allocate a new capability slot.*
- `template<typename T >`  
[L4::Cap< T > alloc \(\)](#) throw ()  
*Allocate a new capability slot.*
- `void` [take \(L4::Cap< void > cap\)](#) throw ()  
*Increase the reference counter for the capability.*
- `bool` [free \(L4::Cap< void > cap, l4\\_cap\\_idx\\_t task=L4\\_INVALID\\_CAP, unsigned unmap\\_flags=L4\\_FP\\_ALL\\_SPACES\)](#) throw ()  
*Free the capability.*
- `bool` [release \(L4::Cap< void > cap, l4\\_cap\\_idx\\_t task=L4\\_INVALID\\_CAP, unsigned unmap\\_flags=L4\\_FP\\_ALL\\_SPACES\)](#) throw ()  
*Decrease the reference counter for a capability.*
- `long` [last \(\)](#) throw ()  
*Return highest capability id managed by this allocator.*

## Protected Member Functions

- [Counting\\_cap\\_alloc \(\)](#) throw ()  
*Create a new, empty allocator.*
- `void` [setup \(void \\*m, long capacity, long bias\)](#) throw ()  
*Set up the backing memory for the allocator and the area of managed capability slots.*

### 14.254.1 Detailed Description

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
class L4Re::Util::Counting_cap_alloc< COUNTERTYPE >
```

Internal reference-counting cap allocator.

This is intended for internal use only. [L4Re](#) applications should use [L4Re::Util::cap\\_alloc\(\)](#).

Allocator for capability slots that automatically frees the slot and optionally unmaps the capability when the reference count goes down to zero. Reference counting must be done manually via [take\(\)](#) and [release\(\)](#). The backing store for the reference counters must be provided in the [setup\(\)](#) method. The allocator can recognize capability slots that are not managed by itself and does nothing on such slots.

#### Note

The user must ensure that the backing store is zero-initialized.

The user must ensure that the capability slots managed by this allocator are not used by a different allocator, see [setup\(\)](#).

The operations in this class are not thread-safe.

Definition at line 75 of file [counting\\_cap\\_alloc](#).

### 14.254.2 Constructor & Destructor Documentation

#### 14.254.2.1 Counting\_cap\_alloc()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::Counting_cap_alloc () throw () [inline],
[protected]
```

Create a new, empty allocator.

Needs to be initialized with [setup\(\)](#) before it can be used.

Definition at line 104 of file [counting\\_cap\\_alloc](#).

### 14.254.3 Member Function Documentation

## 14.254.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4::Cap<void> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc () throw () [inline]
```

Allocate a new capability slot.

**Returns**

The newly allocated capability slot, invalid if the allocator was exhausted.

**Examples:**

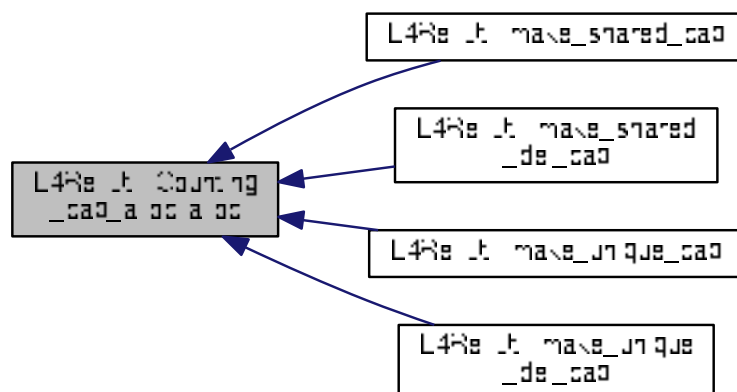
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared+\\_ds/ds\\_srv.cc](#).

Definition at line 135 of file [counting\\_cap\\_alloc](#).

References [L4::Cap\\_base::Invalid](#), and [L4\\_CAP\\_SHIFT](#).

Referenced by [L4Re::Util::make\\_shared\\_cap\(\)](#), [L4Re::Util::make\\_shared\\_del\\_cap\(\)](#), [L4Re::Util::make\\_unique\\_←\\_cap\(\)](#), and [L4Re::Util::make\\_unique\\_del\\_cap\(\)](#).

Here is the caller graph for this function:



### 14.254.3.2 `alloc()` [2/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
template<typename T >
L4::Cap<T> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc () throw () [inline]
```

Allocate a new capability slot.

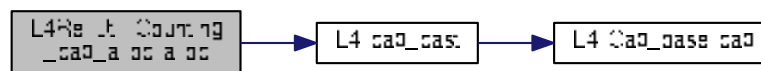
#### Returns

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 156 of file `counting_cap_alloc`.

References [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:



### 14.254.3.3 `free()`

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free (
 L4::Cap< void > cap,
 l4_cap_idx_t task = L4_INVALID_CAP,
 unsigned unmap_flags = L4_FP_ALL_SPACES) throw () [inline]
```

Free the capability.

#### Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <i>cap</i>         | Capability to free.                                  |
| <i>task</i>        | If set, task to unmap the capability from.           |
| <i>unmap_flags</i> | Flags for unmap, see <code>l4_unmap_flags_t</code> . |

#### Precondition

The capability has been allocated. Calling `free` twice on a capability managed by this allocator results in undefined behaviour.



**Returns**

True, if the capability was managed by this allocator.

**Examples:**

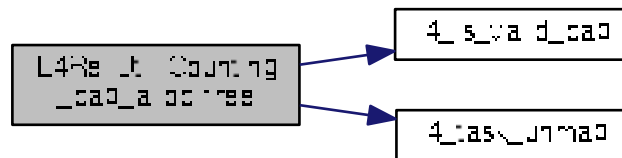
[examples/libs/l4re/c++/mem\\_alloc/main.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 198 of file [counting\\_cap\\_alloc](#).

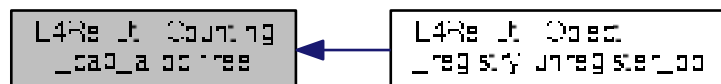
References [l4\\_assert](#), [L4\\_CAP\\_SHIFT](#), [l4\\_is\\_valid\\_cap\(\)](#), and [l4\\_task\\_unmap\(\)](#).

Referenced by [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.254.3.4 release()**

```

template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::release (
 L4::Cap< void > cap,
 l4_cap_idx_t task = L4_INVALID_CAP,
 unsigned unmap_flags = L4_FP_ALL_SPACES) throw () [inline]

```

Decrease the reference counter for a capability.

## Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <i>cap</i>         | Capability to release.                               |
| <i>task</i>        | If set, task to unmap the capability from.           |
| <i>unmap_flags</i> | Flags for unmap, see <code>l4_unmap_flags_t</code> . |

## Precondition

The capability has been allocated. Calling release on a free capability results in undefined behaviour.

## Returns

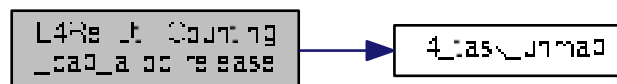
True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or is not managed by this allocator.

Does nothing apart from returning false if the capability is not managed by this allocator.

Definition at line 241 of file `counting_cap_alloc`.

References `l4_assert`, `L4_CAP_SHIFT`, `L4_INVALID_CAP`, and `l4_task_unmap()`.

Here is the call graph for this function:

14.254.3.5 `setup()`

```

template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::CountingCapAlloc< COUNTERTYPE >::setup (
 void * m,
 long capacity,
 long bias) throw () [inline], [protected]

```

Set up the backing memory for the allocator and the area of managed capability slots.

## Parameters

|                 |                                               |
|-----------------|-----------------------------------------------|
| <i>m</i>        | Pointer to backing memory.                    |
| <i>capacity</i> | Number of capabilities that can be stored.    |
| <i>bias</i>     | First capability id to use by this allocator. |

The allocator will manage the capability slots between `bias` and `bias + capacity - 1` (inclusive). It is the responsibility of the user to ensure that these slots are not used otherwise.

Definition at line 121 of file [counting\\_cap\\_alloc](#).

#### 14.254.3.6 take()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::take (
 L4::Cap< void > cap) throw () [inline]
```

Increase the reference counter for the capability.

#### Parameters

|            |                                                          |
|------------|----------------------------------------------------------|
| <i>cap</i> | Capability, whose reference counter should be increased. |
|------------|----------------------------------------------------------|

If the capability was still free, it will be automatically allocated. Silently does nothing if the capability is not managed by this allocator.

Definition at line 171 of file [counting\\_cap\\_alloc](#).

References [L4\\_CAP\\_SHIFT](#).

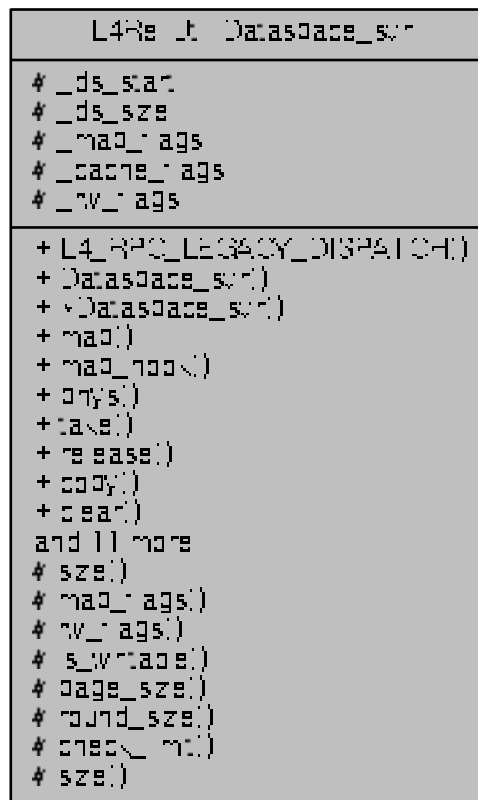
The documentation for this class was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

## 14.255 L4Re::Util::Dataspace\_svr Class Reference

[Dataspace](#) server class.

Collaboration diagram for L4Re::Util::Dataspace\_svr:



## Public Member Functions

- `int map (l4_addr_t offset, l4_addr_t local_addr, unsigned long flags, l4_addr_t min_addr, l4_addr_t max_addr, L4::lpc::Snd_fpage &memory)`  
*Map a region of the dataspace.*
- `virtual int map_hook (l4_addr_t offs, unsigned long flags, l4_addr_t min, l4_addr_t max)`  
*A hook that is called as the first operation in each map request.*
- `virtual int phys (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size) throw ()`  
*Return physical address for a virtual address.*
- `virtual void take () throw ()`  
*Take a reference to this dataspace.*
- `virtual unsigned long release () throw ()`  
*Release a reference to this dataspace.*
- `virtual long copy (l4_addr_t dst_offs, l4_umword_t src_id, l4_addr_t src_offs, unsigned long size) throw ()`  
*Copy from src dataspace to this destination dataspace.*
- `virtual long clear (unsigned long offs, unsigned long size) const throw ()`  
*Clear a region in the dataspace.*
- `virtual long allocate (l4_addr_t offset, l4_size_t size, unsigned access) throw ()`  
*Allocate a region within a dataspace.*

- virtual unsigned long [page\\_shift](#) () const throw ()  
*Define the size of the flexpage to map.*
- virtual bool [is\\_static](#) () const throw ()  
*Return whether the dataspace is static.*

### 14.255.1 Detailed Description

[Dataspace](#) server class.

The default implementation of the interface provides a continuously mapped dataspace.

Definition at line 40 of file [dataspace\\_svr](#).

### 14.255.2 Member Function Documentation

#### 14.255.2.1 allocate()

```
virtual long L4Re::Util::Dataspace_svr::allocate (
 l4_addr_t offset,
 l4_size_t size,
 unsigned access) throw () [inline], [virtual]
```

Allocate a region within a dataspace.

#### Parameters

|               |                                                                                     |
|---------------|-------------------------------------------------------------------------------------|
| <i>offset</i> | Offset in the dataspace, in bytes.                                                  |
| <i>size</i>   | Size of the range, in bytes.                                                        |
| <i>access</i> | Access mode with which the memory backing the dataspace region should be allocated. |

#### Return values

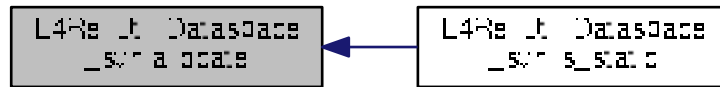
|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 168 of file [dataspace\\_svr](#).

References [L4\\_ENODEV](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



#### 14.255.2.2 clear()

```
virtual long L4Re::Util::Dataspace_svr::clear (
 unsigned long offs,
 unsigned long size) const throw () [virtual]
```

Clear a region in the dataspace.

##### Parameters

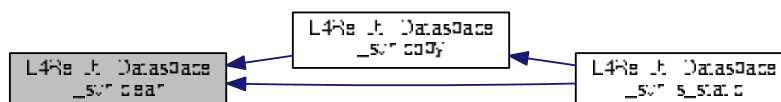
|             |                     |
|-------------|---------------------|
| <i>offs</i> | Start of the region |
| <i>size</i> | Size of the region  |

##### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Referenced by [copy\(\)](#), and [is\\_static\(\)](#).

Here is the caller graph for this function:



## 14.255.2.3 copy()

```
virtual long L4Re::Util::Dataspace_svr::copy (
 l4_addr_t dst_offs,
 l4_umword_t src_id,
 l4_addr_t src_offs,
 unsigned long size) throw () [inline], [virtual]
```

Copy from src dataspace to this destination dataspace.

## Parameters

|                 |                                       |
|-----------------|---------------------------------------|
| <i>dst_offs</i> | Offset into the destination dataspace |
| <i>src_id</i>   | Local id of the source dataspace      |
| <i>src_offs</i> | Offset into the source dataspace      |
| <i>size</i>     | Number of bytes to copy               |

## Return values

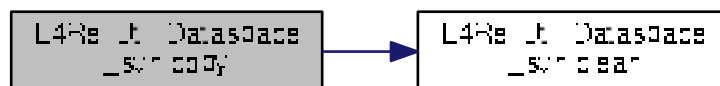
|          |                                                                     |
|----------|---------------------------------------------------------------------|
| $\geq 0$ | Number of bytes copied                                              |
| $< 0$    | An error occurred. The error code may depend on the implementation. |

Definition at line 139 of file [dataspace\\_svr](#).

References [clear\(\)](#), and [L4\\_ENODEV](#).

Referenced by [is\\_static\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.255.2.4 `is_static()`

```
virtual bool L4Re::Util::Dataspace_svr::is_static () const throw () [inline], [virtual]
```

Return whether the dataspace is static.

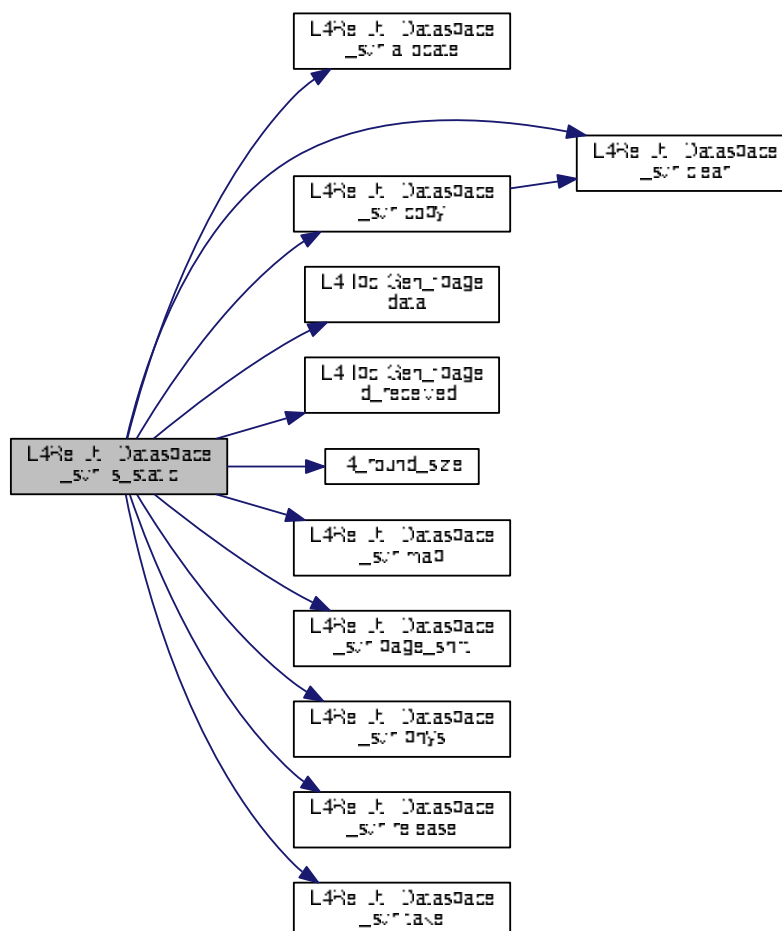
**Returns**

True if dataspace is static

Definition at line 184 of file [dataspace\\_svr](#).

References [allocate\(\)](#), [clear\(\)](#), [copy\(\)](#), [L4::lpc::Gen\\_fpage< T >::data\(\)](#), [L4Re::Dataspace::Stats::flags](#), [L4::lpc::Gen\\_fpage< T >::id\\_received\(\)](#), [L4\\_CAP\\_FPAGE\\_W](#), [L4\\_EACCESS](#), [L4\\_EINVAL](#), [L4\\_EOK](#), [L4\\_EPERM](#), [l4\\_< round\\_size\(\)](#), [map\(\)](#), [page\\_shift\(\)](#), [phys\(\)](#), [release\(\)](#), [L4Re::Dataspace::Stats::size](#), and [take\(\)](#).

Here is the call graph for this function:





## 14.255.2.5 map()

```
int L4Re::Util::Dataspace_svr::map (
 l4_addr_t offset,
 l4_addr_t local_addr,
 unsigned long flags,
 l4_addr_t min_addr,
 l4_addr_t max_addr,
 L4::Ipc::Snd_fpage & memory)
```

Map a region of the dataspace.

## Parameters

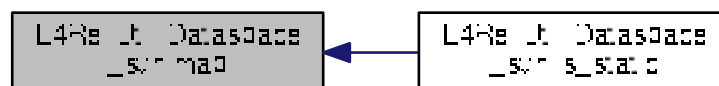
|     |                   |                                                             |
|-----|-------------------|-------------------------------------------------------------|
|     | <i>offset</i>     | Offset to start within data space                           |
|     | <i>local_addr</i> | Local address to map to.                                    |
|     | <i>flags</i>      | Map flags, see <a href="#">L4Re::Dataspace::Map_flags</a> . |
|     | <i>min_addr</i>   | Defines start of receive window.                            |
|     | <i>max_addr</i>   | Defines end of receive window.                              |
| out | <i>memory</i>     | Send fpage to map                                           |

## Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



## 14.255.2.6 map\_hook()

```
virtual int L4Re::Util::Dataspace_svr::map_hook (
 l4_addr_t offs,
 unsigned long flags,
 l4_addr_t min,
 l4_addr_t max) [inline], [virtual]
```

A hook that is called as the first operation in each map request.

## Parameters

|              |                    |
|--------------|--------------------|
| <i>offs</i>  | Offs param to map  |
| <i>flags</i> | Flags param to map |
| <i>min</i>   | Min param to map   |
| <i>max</i>   | Max param to map   |

## Return values

|               |                                                            |
|---------------|------------------------------------------------------------|
| <i>&lt;0</i>  | Error and the map request will be aborted with that error. |
| <i>&gt;=0</i> | Success                                                    |

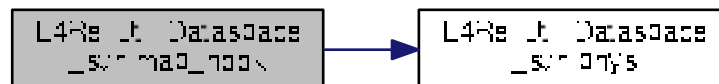
## See also

[map](#)

Definition at line 90 of file [dataspace\\_svr](#).

References [phys\(\)](#).

Here is the call graph for this function:

14.255.2.7 `page_shift()`

```
virtual unsigned long L4Re::Util::Dataspace_svr::page_shift () const throw () [inline],
[virtual]
```

Define the size of the flexpage to map.

**Returns**

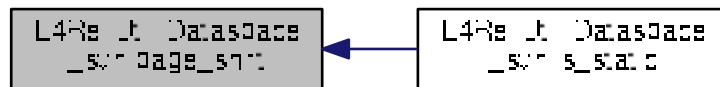
flexpage size

Definition at line 176 of file [dataspace\\_svr](#).

References [L4\\_LOG2\\_PAGESIZE](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:

**14.255.2.8 phys()**

```
virtual int L4Re::Util::Dataspace_svr::phys (
 l4_addr_t offset,
 l4_addr_t & phys_addr,
 l4_size_t & phys_size) throw () [virtual]
```

Return physical address for a virtual address.

**Parameters**

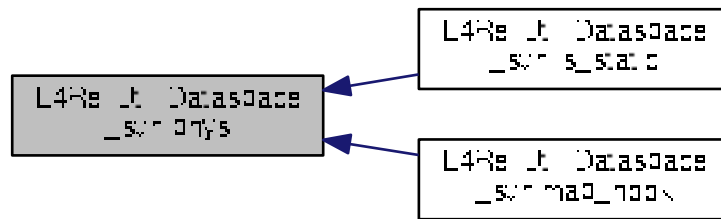
|     |                  |                                   |
|-----|------------------|-----------------------------------|
|     | <i>offset</i>    | Offset into the dataspace         |
| out | <i>phys_addr</i> | Physical address                  |
| out | <i>phys_size</i> | Size of continous physical region |

**Return values**

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Referenced by [is\\_static\(\)](#), and [map\\_hook\(\)](#).

Here is the caller graph for this function:



#### 14.255.2.9 release()

```
virtual unsigned long L4Re::Util::Dataspace_svr::release () throw () [inline], [virtual]
```

Release a reference to this dataspace.

#### Returns

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 124 of file [dataspace\\_svr](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



## 14.255.2.10 take()

```
virtual void L4Re::Util::Dataspace_svr::take () throw () [inline], [virtual]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 114 of file [dataspace\\_svr](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



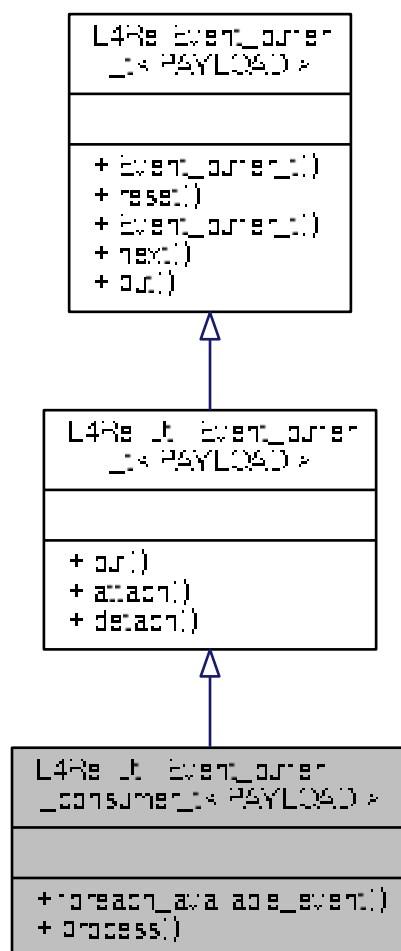
The documentation for this class was generated from the following file:

- [l4/re/util/dataspace\\_svr](#)

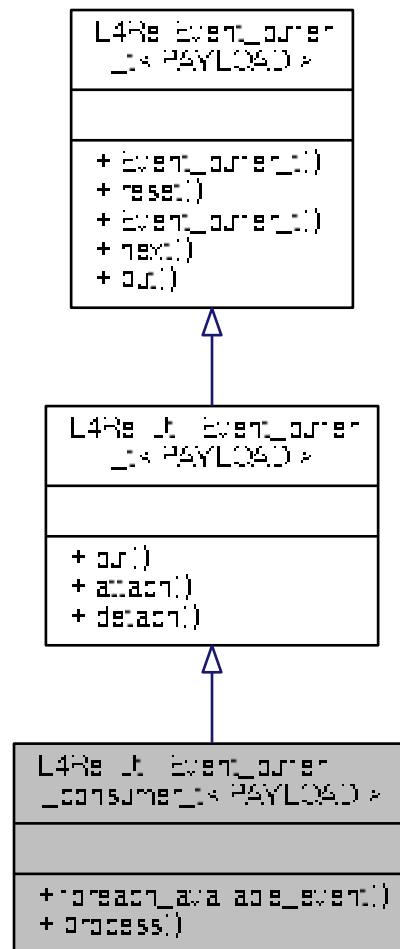
## 14.256 L4Re::Util::Event\_buffer\_consumer\_t&lt; PAYLOAD &gt; Class Template Reference

An event buffer consumer.

Inheritance diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



## Public Member Functions

- `template<typename CB , typename D >`  
`void foreach\_available\_event (CB const &cb, D data=D())`  
*Call function on every available event.*
- `template<typename CB , typename D >`  
`void process (L4::Cap< L4::Irq > irq, L4::Cap< L4::Thread > thread, CB const &cb, D data=D())`  
*Continuously wait for events and process them.*

### 14.256.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_consumer_t< PAYLOAD >
```

An event buffer consumer.

Definition at line 92 of file [event\\_buffer](#).

## 14.256.2 Member Function Documentation

### 14.256.2.1 foreach\_available\_event()

```
template<typename PAYLOAD >
template<typename CB , typename D >
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::foreach_available_event (
 CB const & cb,
 D data = D()) [inline]
```

Call function on every available event.

#### Parameters

|             |                                              |
|-------------|----------------------------------------------|
| <i>cb</i>   | Function callback.                           |
| <i>data</i> | Data to pass as an argument to the callback. |

Definition at line 103 of file [event\\_buffer](#).

References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::free\(\)](#).

Here is the call graph for this function:



### 14.256.2.2 process()

```
template<typename PAYLOAD >
template<typename CB , typename D >
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process (
 L4::Cap< L4::Irq > irq,
 L4::Cap< L4::Thread > thread,
 CB const & cb,
 D data = D()) [inline]
```

Continuously wait for events and process them.

#### Parameters

|               |                                                           |
|---------------|-----------------------------------------------------------|
| <i>irq</i>    | <a href="#">Event</a> signal to wait for.                 |
| <i>thread</i> | Thread capability of the thread calling this function.    |
| <i>cb</i>     | Callback function that is called for each received event. |
| <i>data</i>   | Data to pass as an argument to the processing callback.   |



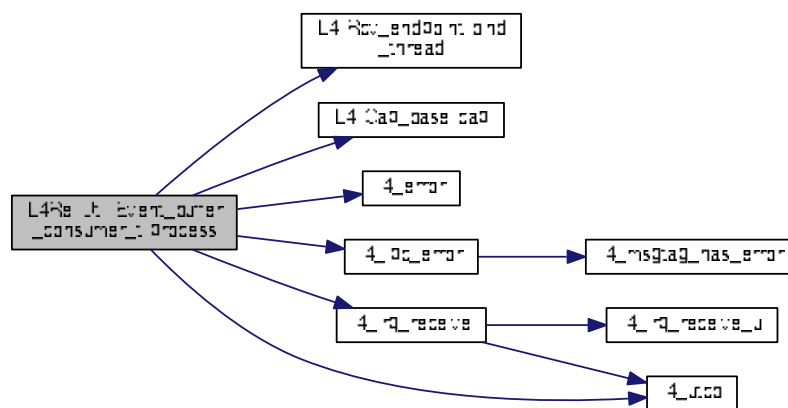
**Note**

This function never returns.

Definition at line 124 of file [event\\_buffer](#).

References [L4::Rcv\\_endpoint::bind\\_thread\(\)](#), [L4::Cap\\_base::cap\(\)](#), [l4\\_error\(\)](#), [l4\\_ipc\\_error\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_irq\\_receive\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



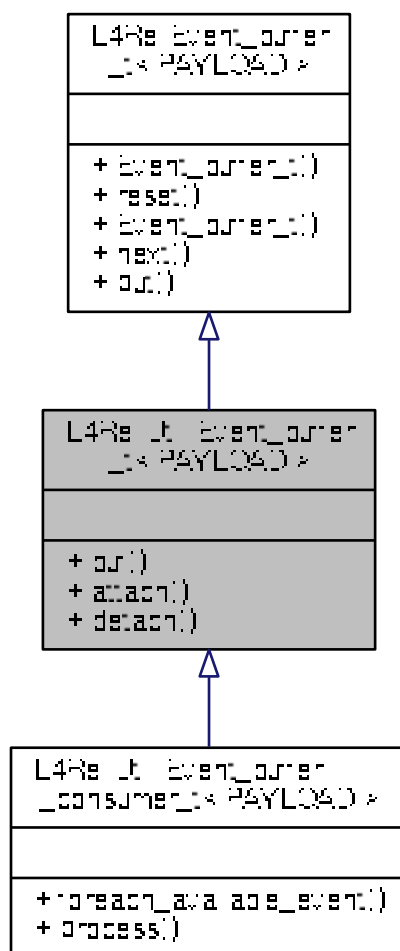
The documentation for this class was generated from the following file:

- `l4/re/util/event_buffer`

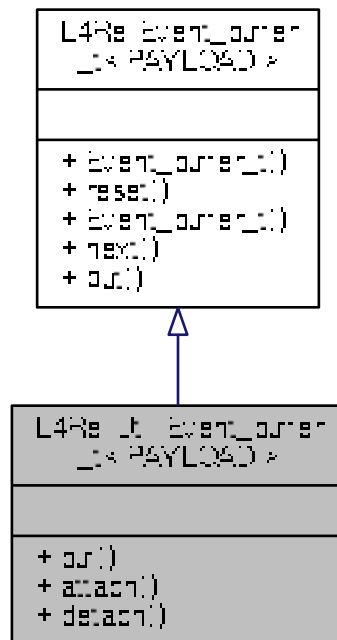
## 14.257 L4Re::Util::Event\_buffer\_t< PAYLOAD > Class Template Reference

Event\_buffer utility class.

Inheritance diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



## Public Member Functions

- void \* [buf](#) () const throw ()  
*Return the buffer.*
- long [attach](#) (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) throw ()  
*Attach event buffer from address space.*
- long [detach](#) (L4::Cap< L4Re::Rm > rm) throw ()  
*Detach event buffer from address space.*

### 14.257.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >
```

Event\_buffer utility class.

Definition at line 36 of file [event\\_buffer](#).

### 14.257.2 Member Function Documentation

### 14.257.2.1 attach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
 L4::Cap< L4Re::Dataspace > ds,
 L4::Cap< L4Re::Rm > rm) throw () [inline]
```

Attach event buffer from address space.

#### Parameters

|           |                                                |
|-----------|------------------------------------------------|
| <i>ds</i> | <a href="#">Dataspace</a> of the event buffer. |
| <i>rm</i> | Region manager to attach buffer to.            |

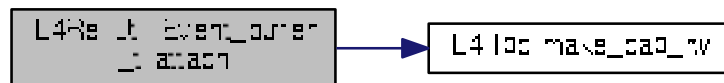
#### Returns

0 on success, negative error code otherwise.

Definition at line 56 of file [event\\_buffer](#).

References [L4::ipc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::Search\\_addr](#).

Here is the call graph for this function:



### 14.257.2.2 buf()

```
template<typename PAYLOAD >
void* L4Re::Util::Event_buffer_t< PAYLOAD >::buf () const throw () [inline]
```

Return the buffer.

#### Returns

Pointer to the event buffer.

Definition at line 46 of file [event\\_buffer](#).

### 14.257.2.3 detach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::detach (
 L4::Cap< L4Re::Rm > rm) throw () [inline]
```

Detach event buffer from address space.

#### Parameters

|           |                                       |
|-----------|---------------------------------------|
| <i>rm</i> | Region manager to detach buffer from. |
|-----------|---------------------------------------|

#### Returns

0 on success, negative error code otherwise.

Definition at line 77 of file [event\\_buffer](#).

The documentation for this class was generated from the following file:

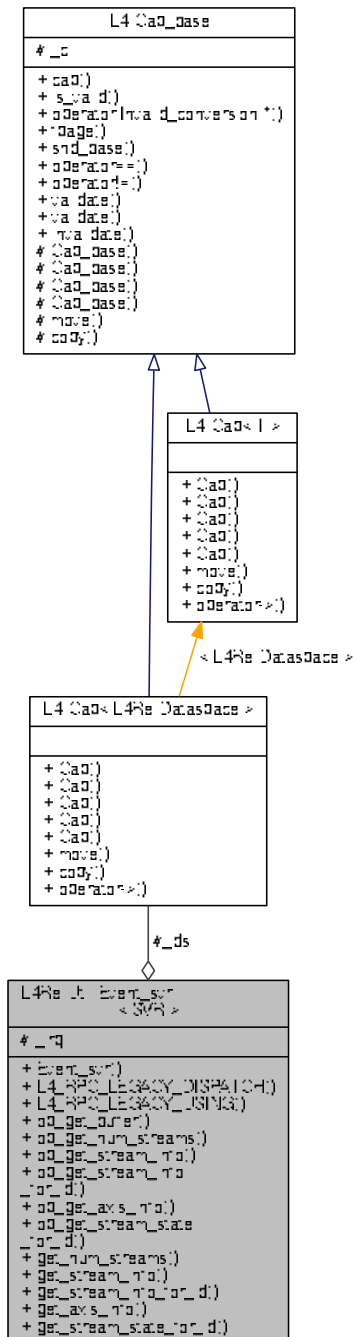
- l4/re/util/event\_buffer

## 14.258 L4Re::Util::Event\_svr< SVR > Class Template Reference

Convenience wrapper for implementing an event server.

Inherits L4Re::Util::Icu\_cap\_array\_svr< ICU >.

Collaboration diagram for L4Re::Util::Event\_svr< SVR >:



## Public Member Functions

- long `op_get_buffer` (L4Re::Event::Rights, [L4::lpc::Cap< L4Re::Dataspace >](#) &ds)  
Handle [L4Re::Event](#) protocol.

### 14.258.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Event_svr< SVR >
```

Convenience wrapper for implementing an event server.

See also

[L4Re::Event](#), [L4Re::Util::Event\\_t](#)

Definition at line 39 of file [event\\_svr](#).

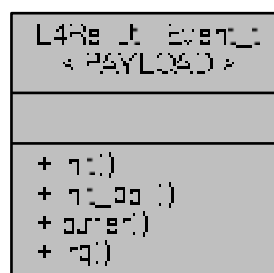
The documentation for this class was generated from the following file:

- [l4/re/util/event\\_svr](#)

## 14.259 L4Re::Util::Event\_t< PAYLOAD > Class Template Reference

Convenience wrapper for getting access to an event object.

Collaboration diagram for L4Re::Util::Event\_t< PAYLOAD >:



### Public Types

- enum [Mode](#) { [Mode\\_irq](#), [Mode\\_polling](#) }
- Modes of operation.*

## Public Member Functions

- `template<typename IRQ_TYPE >`  
`int init (L4::Cap< L4Re::Event > event, L4Re::Env const *env=L4Re::Env::env(), L4Re::Cap_alloc *ca=L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))`  
*Initialise an event object.*
- `int init_poll (L4::Cap< L4Re::Event > event, L4Re::Env const *env=L4Re::Env::env(), L4Re::Cap_alloc *ca=L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))`  
*Initialise an event object in polling mode.*
- `L4Re::Event_buffer_t< PAYLOAD > & buffer ()`  
*Get event buffer.*
- `L4::Cap< L4::Triggerable > irq () const`  
*Get event IRQ.*

### 14.259.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_t< PAYLOAD >
```

Convenience wrapper for getting access to an event object.

After calling `init()` the class supplies the event-buffer and the associated IRQ object.

Definition at line 43 of file `event`.

### 14.259.2 Member Enumeration Documentation

#### 14.259.2.1 Mode

```
template<typename PAYLOAD >
enum L4Re::Util::Event_t::Mode
```

Modes of operation.

Enumerator

|              |                                                 |
|--------------|-------------------------------------------------|
| Mode_irq     | Create an IRQ and attach, to get notifications. |
| Mode_polling | Do not use an IRQ.                              |

Definition at line 49 of file `event`.

### 14.259.3 Member Function Documentation



## 14.259.3.1 buffer()

```
template<typename PAYLOAD >
L4Re::Event_buffer_t<PAYLOAD>& L4Re::Util::Event_t< PAYLOAD >::buffer () [inline]
```

Get event buffer.

## Returns

[Event](#) buffer object.

Definition at line 157 of file [event](#).

## 14.259.3.2 init()

```
template<typename PAYLOAD >
template<typename IRQ_TYPE >
int L4Re::Util::Event_t< PAYLOAD >::init (
 L4::Cap< L4Re::Event > event,
 L4Re::Env const * env = L4Re::Env::env(),
 L4Re::Cap_alloc * ca = L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc))
[inline]
```

Initialise an event object.

## Template Parameters

|                 |                                                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>IRQ_TYPE</i> | Type used for handling notifications from the event provider. This must be derived from <a href="#">L4::Triggerable</a> . |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|

## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>event</i> | Capability to event.             |
| <i>env</i>   | Pointer to L4Re-Environment      |
| <i>ca</i>    | Pointer to capability allocator. |

## Return values

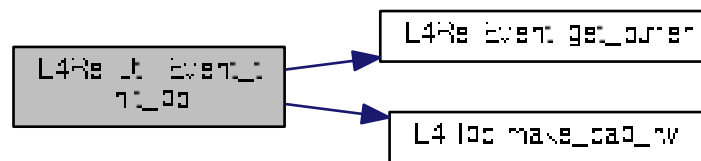
|            |                                              |
|------------|----------------------------------------------|
| 0          | Success                                      |
| -L4_ENOMEM | No memory to allocate required capabilities. |
| <0         | Other IPC errors.                            |

Definition at line 70 of file [event](#).

References [L4::Icu::bind\(\)](#), [L4Re::Event::get\\_buffer\(\)](#), [L4\\_ENOMEM](#), [l4\\_error\(\)](#), [L4::lpc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::Search\\_addr](#).



Here is the call graph for this function:



#### 14.259.3.4 irq()

```
template<typename PAYLOAD >
L4::Cap<L4::Triggerable> L4Re::Util::Event_t< PAYLOAD >::irq () const [inline]
```

Get event IRQ.

Returns

Event IRQ.

Definition at line 164 of file [event](#).

The documentation for this class was generated from the following file:

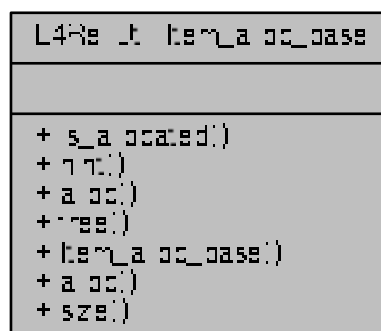
- [l4/re/util/event](#)

## 14.260 L4Re::Util::Item\_alloc\_base Class Reference

Item allocator.

Inherited by `L4Re::Util::Item_alloc< Bits >`.

Collaboration diagram for `L4Re::Util::Item_alloc_base`:



### 14.260.1 Detailed Description

Item allocator.

Definition at line 38 of file [item\\_alloc](#).

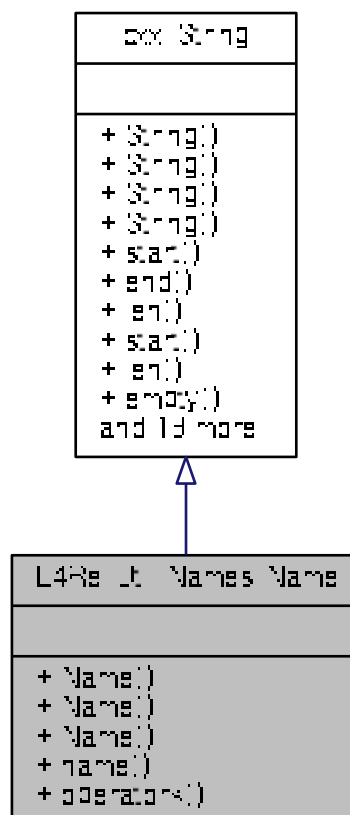
The documentation for this class was generated from the following file:

- [l4/re/util/item\\_alloc](#)

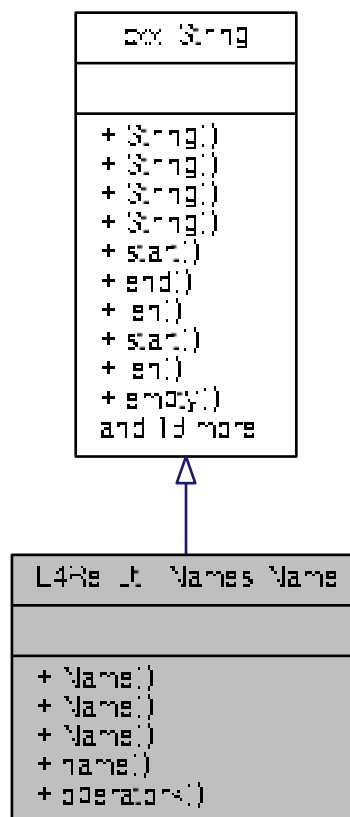
### 14.261 L4Re::Util::Names::Name Class Reference

[Name](#) class.

Inheritance diagram for L4Re::Util::Names::Name:



Collaboration diagram for L4Re::Util::Names::Name:



## Additional Inherited Members

### 14.261.1 Detailed Description

[Name](#) class.

Definition at line 42 of file [name\\_space\\_svr](#).

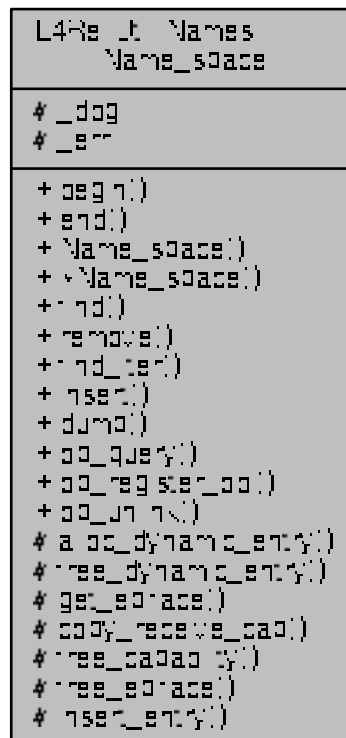
The documentation for this class was generated from the following file:

- [l4/re/util/name\\_space\\_svr](#)

## 14.262 L4Re::Util::Names::Name\_space Class Reference

Abstract server-side implementation of the `L4::Namespace` interface.

Collaboration diagram for L4Re::Util::Names::Name\_space:



## Protected Member Functions

- virtual Entry \* [alloc\\_dynamic\\_entry](#) (Name const &n, unsigned flags)=0  
*Allocate a new entry for the given name.*
- virtual void [free\\_dynamic\\_entry](#) (Entry \*e)=0  
*Free an entry previously allocated with [alloc\\_dynamic\\_entry](#)().*
- virtual int [get\\_epiface](#) (l4\_umword\_t data, bool is\_local, L4::Epiface \*\*lo)=0  
*Return a pointer to the epiface assigned to a given label.*
- virtual int [copy\\_receive\\_cap](#) (L4::Cap< void > \*cap)=0  
*Return the receive capability for permanent use.*
- virtual void [free\\_capability](#) (L4::Cap< void > cap)=0  
*Free a capability previously acquired with [copy\\_receive\\_cap](#)().*
- virtual void [free\\_epiface](#) (L4::Epiface \*epiface)=0  
*Free epiface previously acquired with [get\\_epiface](#)().*

### 14.262.1 Detailed Description

Abstract server-side implementation of the L4::Namespace interface.

Definition at line 184 of file [name\\_space\\_svr](#).

## 14.262.2 Member Function Documentation

### 14.262.2.1 alloc\_dynamic\_entry()

```
virtual Entry* L4Re::Util::Names::Name_space::alloc_dynamic_entry (
 Name const & n,
 unsigned flags) [protected], [pure virtual]
```

Allocate a new entry for the given name.

#### Parameters

|              |                                    |
|--------------|------------------------------------|
| <i>n</i>     | Name of the entry, must be copied. |
| <i>flags</i> | Entry flags, see Obj::Flags.       |

#### Returns

A pointer to the newly allocated entry or NULL on error.

This method is called when a new entry was received. It must allocate memory, copy *n* out of the receive buffer and wrap everything in an Entry.

### 14.262.2.2 copy\_receive\_cap()

```
virtual int L4Re::Util::Names::Name_space::copy_receive_cap (
 L4::Cap< void > * cap) [protected], [pure virtual]
```

Return the receive capability for permanent use.

#### Parameters

|     |            |                                                                                                                                |
|-----|------------|--------------------------------------------------------------------------------------------------------------------------------|
| out | <i>cap</i> | Capability slot with the received capability. Must be permanently available until <a href="#">free_capability()</a> is called. |
|-----|------------|--------------------------------------------------------------------------------------------------------------------------------|

This method is called when a new entry is registered together with a capability mapping. It must decide whether and where to store the capability and return the final capability slot. Typical implementations return the capability slot in the receive window and allocate a new receive window.

### 14.262.2.3 free\_capability()

```
virtual void L4Re::Util::Names::Name_space::free_capability (
 L4::Cap< void > cap) [protected], [pure virtual]
```

Free a capability previously acquired with [copy\\_receive\\_cap\(\)](#).

## Parameters

|    |            |                     |
|----|------------|---------------------|
| in | <i>cap</i> | Capability to free. |
|----|------------|---------------------|

Counterpart of [copy\\_receive\\_cap](#). Free the capability slot when the entry is deleted or changed.

## 14.262.2.4 free\_dynamic\_entry()

```
virtual void L4Re::Util::Names::Name_space::free_dynamic_entry (
 Entry * e) [protected], [pure virtual]
```

Free an entry previously allocated with [alloc\\_dynamic\\_entry\(\)](#).

## Parameters

|          |                |
|----------|----------------|
| <i>e</i> | Entry to free. |
|----------|----------------|

## 14.262.2.5 free\_epiface()

```
virtual void L4Re::Util::Names::Name_space::free_epiface (
 L4::Epiface * epiface) [protected], [pure virtual]
```

Free epiface previously acquired with [get\\_epiface\(\)](#).

## Parameters

|    |                |                  |
|----|----------------|------------------|
| in | <i>epiface</i> | Epiface to free. |
|----|----------------|------------------|

Called when an entry that points to an epiface is deleted allowing implementations that hold resources to free them.

## 14.262.2.6 get\_epiface()

```
virtual int L4Re::Util::Names::Name_space::get_epiface (
 l4_umword_t data,
 bool is_local,
 L4::Epiface ** lo) [protected], [pure virtual]
```

Return a pointer to the epiface assigned to a given label.

## Parameters

|     |                 |                                                                                                |
|-----|-----------------|------------------------------------------------------------------------------------------------|
| in  | <i>data</i>     | Label or in the local case the capability slot of the receiving capability.                    |
| in  | <i>is_local</i> | If true, a local capability slot was supplied, if false the label of a locally bound IPC gate. |
| out | <i>lo</i>       | Pointer to epiface responsible for the capability.                                             |



## Returns

[L4\\_EOK](#) if a valid interface could be found or an error message otherwise.

This method is called when a new entry is registered and some local ID was received for the capability. In this case, the generic implementation needs to get the epiface in order to get the capability.

The callee must make sure that the epiface remains valid until `free_epiface` is called. In particular, the capability slot must not be reallocated as long as the namespace server holds a reference to the epiface.

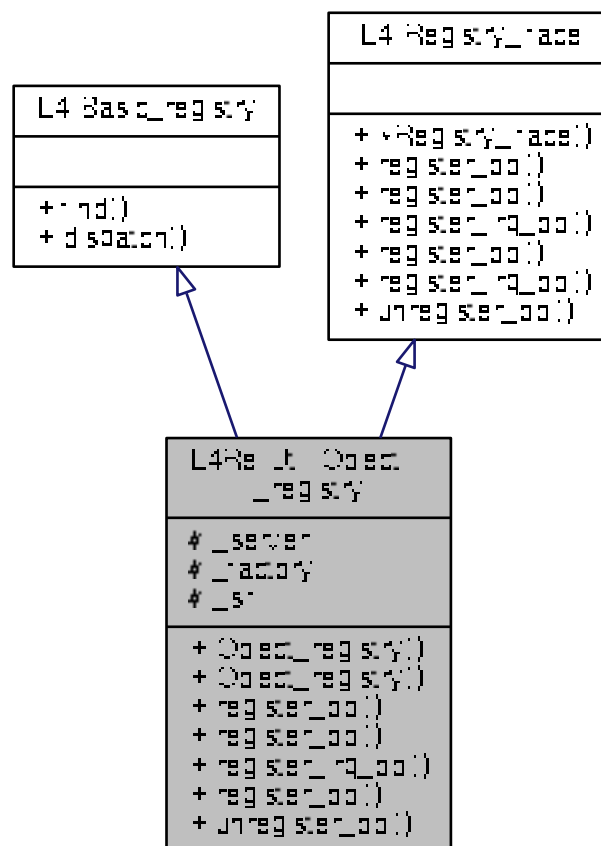
The documentation for this class was generated from the following file:

- `l4/re/util/name_space_svr`

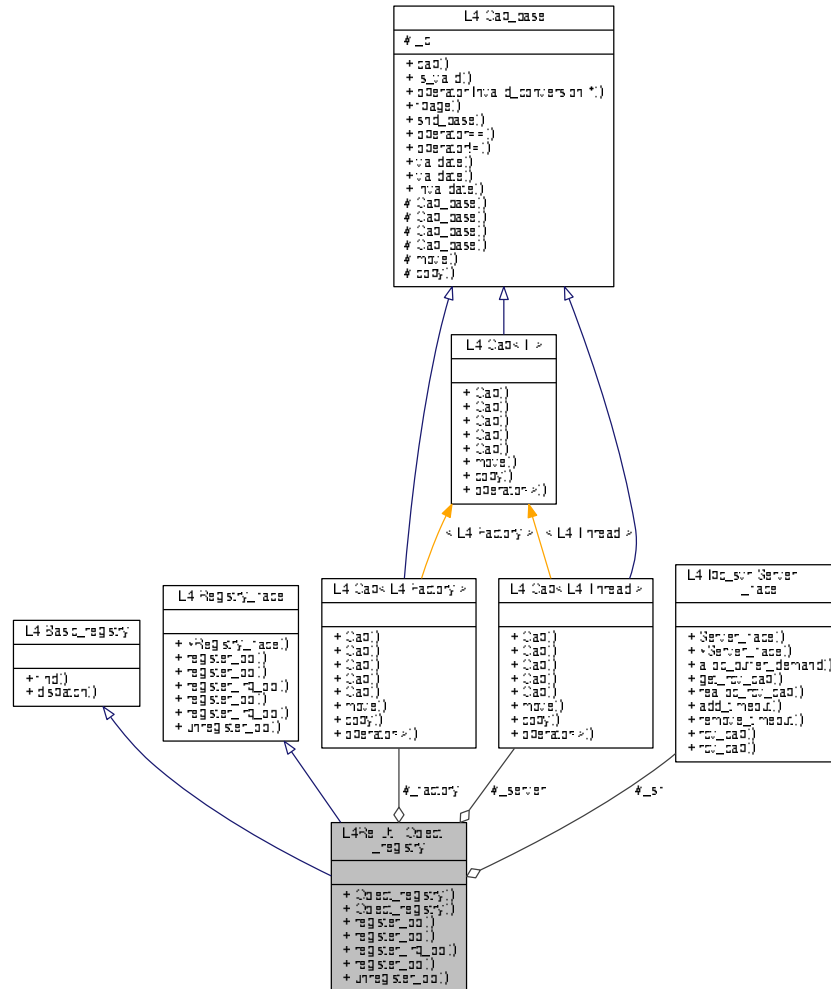
## 14.263 L4Re::Util::Object\_registry Class Reference

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

Inheritance diagram for L4Re::Util::Object\_registry:



Collaboration diagram for L4Re::Util::Object\_registry:



## Public Member Functions

- [Object\\_registry](#) ([L4::lpc\\_svr::Server\\_iface](#) \*sif)  
Create a registry for the main thread of the task using the default factory.
- [Object\\_registry](#) ([L4::lpc\\_svr::Server\\_iface](#) \*sif, [L4::Cap](#)< [L4::Thread](#) > server, [L4::Cap](#)< [L4::Factory](#) > factory)  
Create a registry for arbitrary threads.
- [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o, char const \*service) override  
Register a new server object to a pre-allocated receive endpoint.
- [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o) override  
Register a new server object on a newly allocated capability.
- [L4::Cap](#)< [L4::Irq](#) > [register\\_irq\\_obj](#) ([L4::Epiface](#) \*o) override  
Register a handler for an interrupt.
- [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > [register\\_obj](#) ([L4::Epiface](#) \*o, [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > ep) override  
Register a handler for an already existing interrupt.
- void [unregister\\_obj](#) ([L4::Epiface](#) \*o, bool unmap=true) override  
Remove a server object from the handler list.

## Additional Inherited Members

### 14.263.1 Detailed Description

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

This class manages most of the setup of a server object. If necessary, an IPC gate is created, the specified thread is bound to the IPC gate. Incoming IPC is dispatched to the server object based on the label of the IPC gate.

The object registry is also able to manage IRQ endpoints. They require a different method for the object creation. Otherwise they are handled in the same way as IPC gates: a server object is responsible to process the incoming interrupts.

Definition at line 52 of file [object\\_registry](#).

### 14.263.2 Constructor & Destructor Documentation

#### 14.263.2.1 Object\_registry() [1/2]

```
L4Re::Util::Object_registry::Object_registry (
 L4::Ipc_svr::Server_iface * sif) [inline], [explicit]
```

Create a registry for the main thread of the task using the default factory.

##### Parameters

|            |                        |
|------------|------------------------|
| <i>sif</i> | Server loop interface. |
|------------|------------------------|

Definition at line 78 of file [object\\_registry](#).

#### 14.263.2.2 Object\_registry() [2/2]

```
L4Re::Util::Object_registry::Object_registry (
 L4::Ipc_svr::Server_iface * sif,
 L4::Cap< L4::Thread > server,
 L4::Cap< L4::Factory > factory) [inline]
```

Create a registry for arbitrary threads.

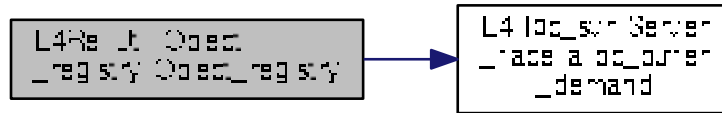
##### Parameters

|                |                                                                   |
|----------------|-------------------------------------------------------------------|
| <i>sif</i>     | Server loop interface.                                            |
| <i>server</i>  | Capability to the thread that executes the server objects.        |
| <i>factory</i> | Capability to a factory object capable of creating new IPC gates. |

Definition at line 92 of file [object\\_registry](#).

References [L4::lpc\\_svr::Server\\_iface::alloc\\_buffer\\_demand\(\)](#).

Here is the call graph for this function:



### 14.263.3 Member Function Documentation

#### 14.263.3.1 register\_irq\_obj()

```
L4::Cap<L4::Irq> L4Re::Util::Object_registry::register_irq_obj (
 L4::Epiface * o) [inline], [override], [virtual]
```

Register a handler for an interrupt.

##### Parameters

|          |                                  |
|----------|----------------------------------|
| <i>o</i> | Server object that handles IRQs. |
|----------|----------------------------------|

##### Returns

The capability the server object was registered with.

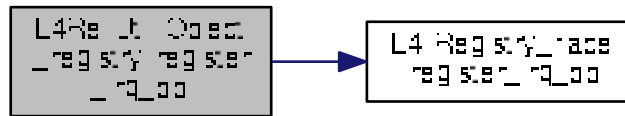
The IRQ will be newly allocated using the registry's factory object.

Implements [L4::Registry\\_iface](#).

Definition at line 229 of file [object\\_registry](#).

References [L4::Registry\\_iface::register\\_irq\\_obj\(\)](#).

Here is the call graph for this function:



### 14.263.3.2 register\_obj() [1/3]

```
L4::Cap<void> L4Re::Util::Object_registry::register_obj (
 L4::Epiface * o,
 char const * service) [inline], [override], [virtual]
```

Register a new server object to a pre-allocated receive endpoint.

#### Parameters

|                |                                           |
|----------------|-------------------------------------------|
| <i>o</i>       | Server object that handles IPC requests.  |
| <i>service</i> | Name of a pre-allocated receive endpoint. |

#### Returns

The capability the server object was registered with.

Implements [L4::Registry\\_iface](#).

Definition at line 201 of file [object\\_registry](#).

### 14.263.3.3 register\_obj() [2/3]

```
L4::Cap<void> L4Re::Util::Object_registry::register_obj (
 L4::Epiface * o) [inline], [override], [virtual]
```

Register a new server object on a newly allocated capability.

#### Parameters

|          |                                          |
|----------|------------------------------------------|
| <i>o</i> | Server object that handles IPC requests. |
|----------|------------------------------------------|

**Returns**

The capability the server object was registered with.

The IPC gate will be allocated using the registry's factory.

Implements [L4::Registry\\_iface](#).

Definition at line 215 of file [object\\_registry](#).

**14.263.3.4 register\_obj()** [3/3]

```
L4::Cap<L4::Rcv_endpoint> L4Re::Util::Object_registry::register_obj (
 L4::Epiface * o,
 L4::Cap< L4::Rcv_endpoint > ep) [inline], [override], [virtual]
```

Register a handler for an already existing interrupt.

**Parameters**

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| <i>o</i>  | Server object that handles the IPC.                                                       |
| <i>ep</i> | Capability to a receive endpoint, may be a hardware or software interrupt or an IPC gate. |

**Returns**

The capability the server object was registered with.

Implements [L4::Registry\\_iface](#).

Definition at line 247 of file [object\\_registry](#).

**14.263.3.5 unregister\_obj()**

```
void L4Re::Util::Object_registry::unregister_obj (
 L4::Epiface * o,
 bool unmap = true) [inline], [override], [virtual]
```

Remove a server object from the handler list.

**Parameters**

|              |                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------|
| <i>o</i>     | Server object to unbind.                                                                                           |
| <i>unmap</i> | Specifies if the object capability shall be unmapped (true) or not. The default (true) is to unmap the capability. |

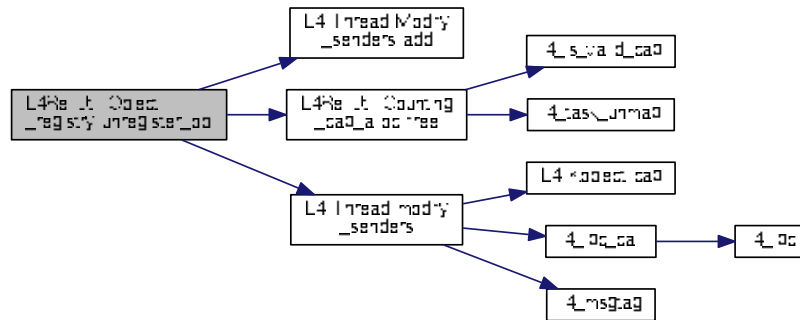
The capability used by the server object will be unmapped if `unmap` is true.

Implements [L4::Registry\\_iface](#).

Definition at line 263 of file [object\\_registry](#).

References [L4::Thread::Modify\\_senders::add\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >::free\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), and [L4::Thread::modify\\_senders\(\)](#).

Here is the call graph for this function:



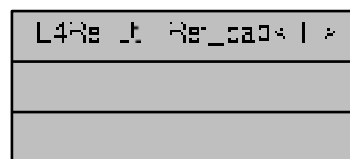
The documentation for this class was generated from the following file:

- [l4/re/util/object\\_registry](#)

## 14.264 L4Re::Util::Ref\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

Collaboration diagram for L4Re::Util::Ref\_cap< T >:



### 14.264.1 Detailed Description

```

template<typename T>
struct L4Re::Util::Ref_cap< T >

```

Automatic capability that implements automatic free and unmap of the capability selector.

## Template Parameters

|          |                                                        |
|----------|--------------------------------------------------------|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--------------------------------------------------------|

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
 L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
 // reference count for the allocated cap selector is now 1

 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

 global_ds_cap = ds_cap;
 // reference count is now 2
 ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 235 of file [cap\\_alloc](#).

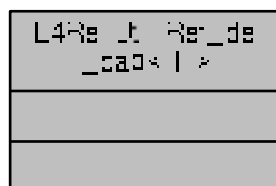
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 14.265 L4Re::Util::Ref\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Collaboration diagram for L4Re::Util::Ref\_del\_cap< T >:



### 14.265.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_del_cap< T >
```

Automatic capability that implements automatic free and unmap+delete of the capability selector.



## Template Parameters

|          |                                                        |
|----------|--------------------------------------------------------|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--------------------------------------------------------|

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Ref\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

## Usage:

```
L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
 L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
 // reference count for the allocated cap selector is now 1

 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

 global_ds_cap = ds_cap;
 // reference count is now 2
 ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 276 of file [cap\\_alloc](#).

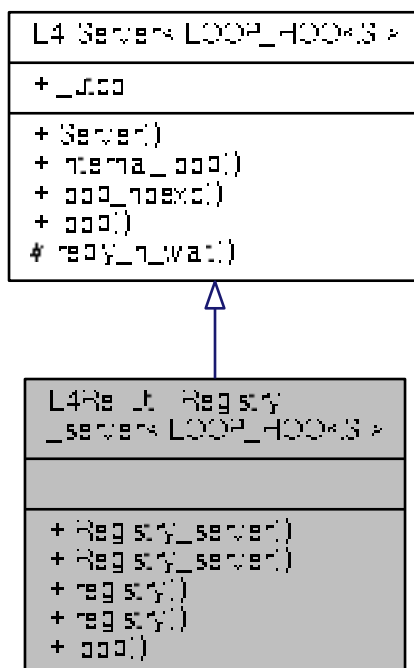
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

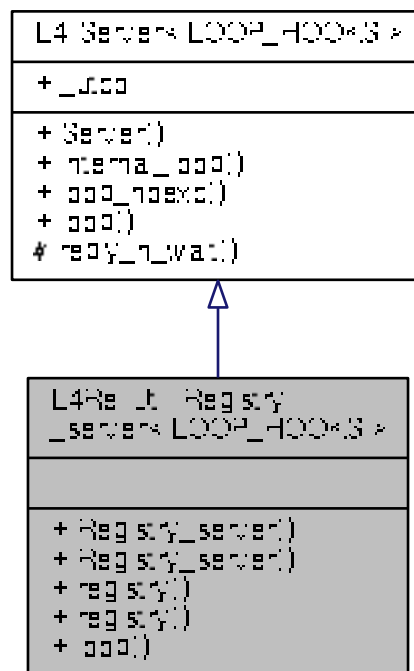
## 14.266 L4Re::Util::Registry\_server&lt; LOOP\_HOOKS &gt; Class Template Reference

A server loop object which has a [Object\\_registry](#) included.

Inheritance diagram for L4Re::Util::Registry\_server< LOOP\_HOOKS >:



Collaboration diagram for L4Re::Util::Registry\_server< LOOP\_HOOKS >:



## Public Member Functions

- [Registry\\_server\(\)](#)  
Create a new server loop object for the main thread of the task.
- [Registry\\_server\(l4\\_utcb\\_t \\*utcb, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory\)](#)  
Create a new server loop object for an arbitrary thread and factory.
- [Object\\_registry](#) const \* [registry\(\)](#) const  
Return registry of this server loop.
- [Object\\_registry](#) \* [registry\(\)](#)  
Return registry of this server loop.
- void [L4\\_NORETURN loop\(\)](#)  
Start the server loop.

## Additional Inherited Members

### 14.266.1 Detailed Description

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
class L4Re::Util::Registry_server< LOOP_HOOKS >
```

A server loop object which has a [Object\\_registry](#) included.

Examples:

[examples/clntsrv/server.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 293 of file [object\\_registry](#).

## 14.266.2 Constructor & Destructor Documentation

### 14.266.2.1 Registry\_server() [1/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server () [inline]
```

Create a new server loop object for the main thread of the task.

#### Precondition

Must be called from the main thread or behaviour is undefined.

Definition at line 305 of file [object\\_registry](#).

### 14.266.2.2 Registry\_server() [2/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
 l4_utcb_t * utcb,
 L4::Cap< L4::Thread > server,
 L4::Cap< L4::Factory > factory) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

#### Parameters

|                |                                                            |
|----------------|------------------------------------------------------------|
| <i>utcb</i>    | The UTCB of the thread running the server loop.            |
| <i>server</i>  | Capability to thread running the server loop.              |
| <i>factory</i> | Capability to factory object used to create new IPC gates. |

Definition at line 315 of file [object\\_registry](#).

## 14.266.3 Member Function Documentation

## 14.266.3.1 registry() [1/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
Object_registry const* L4Re::Util::Registry_server< LOOP_HOOKS >::registry () const [inline]
```

Return registry of this server loop.

Definition at line 321 of file [object\\_registry](#).

## 14.266.3.2 registry() [2/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
Object_registry* L4Re::Util::Registry_server< LOOP_HOOKS >::registry () [inline]
```

Return registry of this server loop.

Definition at line 323 of file [object\\_registry](#).

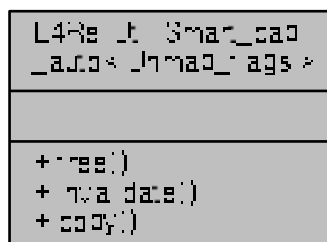
The documentation for this class was generated from the following file:

- [l4/re/util/object\\_registry](#)

## 14.267 L4Re::Util::Smart\_cap\_auto&lt; Unmap\_flags &gt; Class Template Reference

Helper for [Auto\\_cap](#) and [Auto\\_del\\_cap](#).

Collaboration diagram for L4Re::Util::Smart\_cap\_auto< Unmap\_flags >:



## Static Public Member Functions

- static void [free](#) (L4::Cap\_base &c)  
*Free operation for L4::Smart\_cap.*
- static void [invalidate](#) (L4::Cap\_base &c)  
*Invalidate operation for L4::Smart\_cap.*
- static L4::Cap\_base [copy](#) (L4::Cap\_base const &src)  
*Copy operation for L4::Smart\_cap.*

### 14.267.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_cap_auto< Unmap_flags >
```

Helper for [Auto\\_cap](#) and [Auto\\_del\\_cap](#).

Definition at line 59 of file [cap\\_alloc](#).

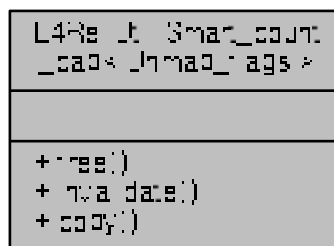
The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 14.268 L4Re::Util::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

Collaboration diagram for L4Re::Util::Smart\_count\_cap< Unmap\_flags >:



### Static Public Member Functions

- static void [free](#) ([L4::Cap\\_base](#) &c) throw ()  
*Free operation for [L4::Smart\\_cap](#) (decrement ref count and delete if 0).*
- static void [invalidate](#) ([L4::Cap\\_base](#) &c) throw ()  
*Invalidate operation for [L4::Smart\\_cap](#).*
- static [L4::Cap\\_base](#) [copy](#) ([L4::Cap\\_base](#) const &src)  
*Copy operation for [L4::Smart\\_cap](#) (increment ref count).*

### 14.268.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_count_cap< Unmap_flags >
```

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

Definition at line 99 of file [cap\\_alloc](#).

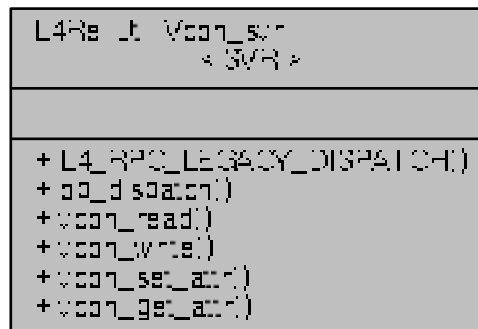
The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 14.269 L4Re::Util::Vcon\_svr< SVR > Class Template Reference

[Console](#) server template class.

Collaboration diagram for L4Re::Util::Vcon\_svr< SVR >:



### 14.269.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Vcon_svr< SVR >
```

[Console](#) server template class.

This template uses `vcon_write()` and `vcon_read()` to get and deliver data from the implementor.

`vcon_read()` needs to update the status argument with the `L4_vcon_read_stat` flags, especially the `L4_VCON_READ_STAT_DONE` flag to indicate that there's nothing more to read for the other end.

`vcon_write()` gets the live data from the UTCB. Make sure to copy out the data before using the UTCB again.

The size parameter of both functions is given in bytes.

Definition at line 46 of file [vcon\\_svr](#).

The documentation for this class was generated from the following file:

- `l4/re/util/vcon_svr`





- Return framebuffer memory dataspace.*
- [L4Re::Video::Goos::Info](#) const \* [screen\\_info](#) () const  
*Goos information structure.*
- [L4Re::Video::View::Info](#) const \* [view\\_info](#) () const  
*View information structure.*
- virtual int [refresh](#) (int x, int y, int w, int h)  
*Refresh area of the framebuffer.*
- void [init\\_infos](#) ()  
*Initialize the view information structure of this object.*
- virtual [~Goos\\_svr](#) ()  
*Destructor.*

## Protected Attributes

- [L4::Cap< L4Re::Dataspace > \\_fb\\_ds](#)  
*Goos memory dataspace.*
- [L4Re::Video::Goos::Info \\_screen\\_info](#)  
*Goos information.*
- [L4Re::Video::View::Info \\_view\\_info](#)  
*View information.*

### 14.270.1 Detailed Description

Goos server class.

Definition at line 36 of file [goos\\_svr](#).

### 14.270.2 Member Function Documentation

#### 14.270.2.1 [get\\_fb\(\)](#)

```
L4::Cap<L4Re::Dataspace> L4Re::Util::Video::Goos_svr::get_fb () const [inline]
```

Return framebuffer memory dataspace.

#### Returns

Goos memory dataspace

Definition at line 53 of file [goos\\_svr](#).

References [\\_fb\\_ds](#).

### 14.270.2.2 init\_infos()

```
void L4Re::Util::Video::Goos_svr::init_infos () [inline]
```

Initialize the view information structure of this object.

This function initializes the view info structure of this goos object based on the information in the goos information, i.e. the width, height and pixel\_info of the goos information has to contain valid values before calling init\_info().

Definition at line 89 of file [goos\\_svr](#).

References [L4Re::Video::View::Info::buffer\\_index](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::Goos::Info::height](#), [L4Re::Video::View::Info::pixel\\_info](#), [L4Re::Video::Goos::Info::pixel\\_info](#), [L4Re::Video::View::Info::view\\_index](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::Goos::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

### 14.270.2.3 refresh()

```
virtual int L4Re::Util::Video::Goos_svr::refresh (
 int x,
 int y,
 int w,
 int h) [inline], [virtual]
```

Refresh area of the framebuffer.

#### Parameters

|          |                          |
|----------|--------------------------|
| <i>x</i> | X coordinate (pixels)    |
| <i>y</i> | Y coordinate (pixels)    |
| <i>w</i> | Width of area in pixels  |
| <i>h</i> | Height of area in pixels |

#### Returns

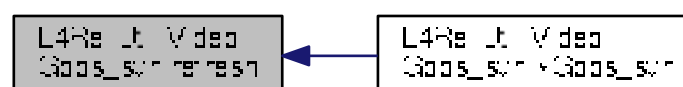
0 on success, negative error code otherwise

Definition at line 77 of file [goos\\_svr](#).

References [L4\\_ENOSYS](#).

Referenced by [~Goos\\_svr\(\)](#).

Here is the caller graph for this function:



#### 14.270.2.4 screen\_info()

```
L4Re::Video::Goos::Info const* L4Re::Util::Video::Goos_svr::screen_info () const [inline]
```

Goos information structure.

##### Returns

Return goos information structure.

Definition at line 59 of file [goos\\_svr](#).

References [\\_screen\\_info](#).

#### 14.270.2.5 view\_info()

```
L4Re::Video::View::Info const* L4Re::Util::Video::Goos_svr::view_info () const [inline]
```

View information structure.

##### Returns

Return view information structure.

Definition at line 65 of file [goos\\_svr](#).

References [\\_view\\_info](#).

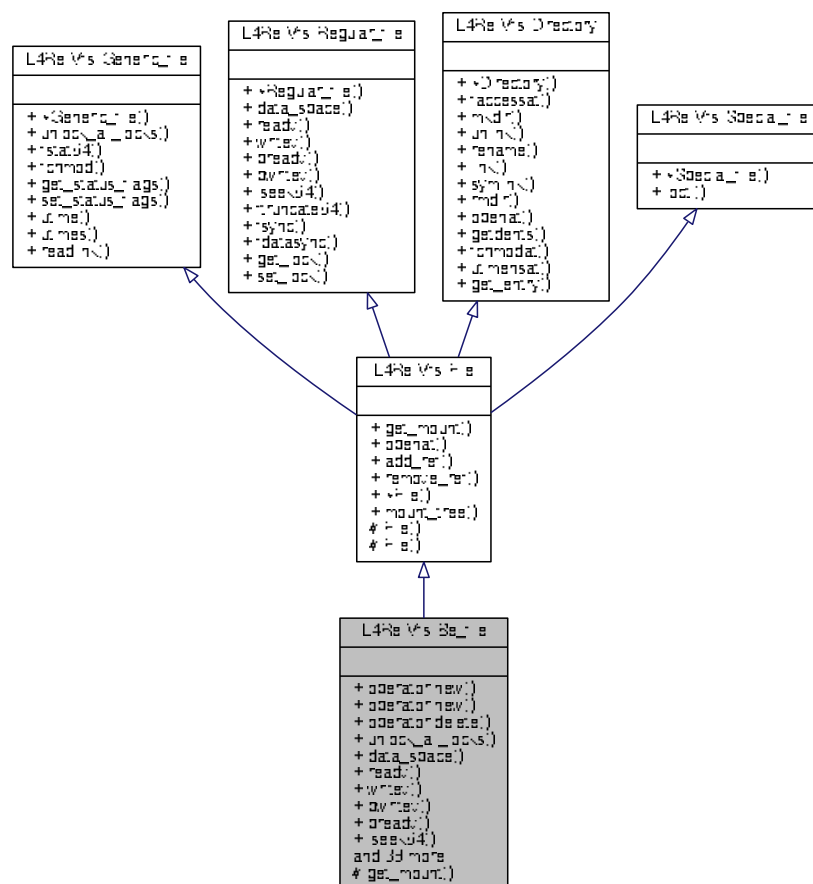
The documentation for this class was generated from the following file:

- [l4/re/util/video/goos\\_svr](#)

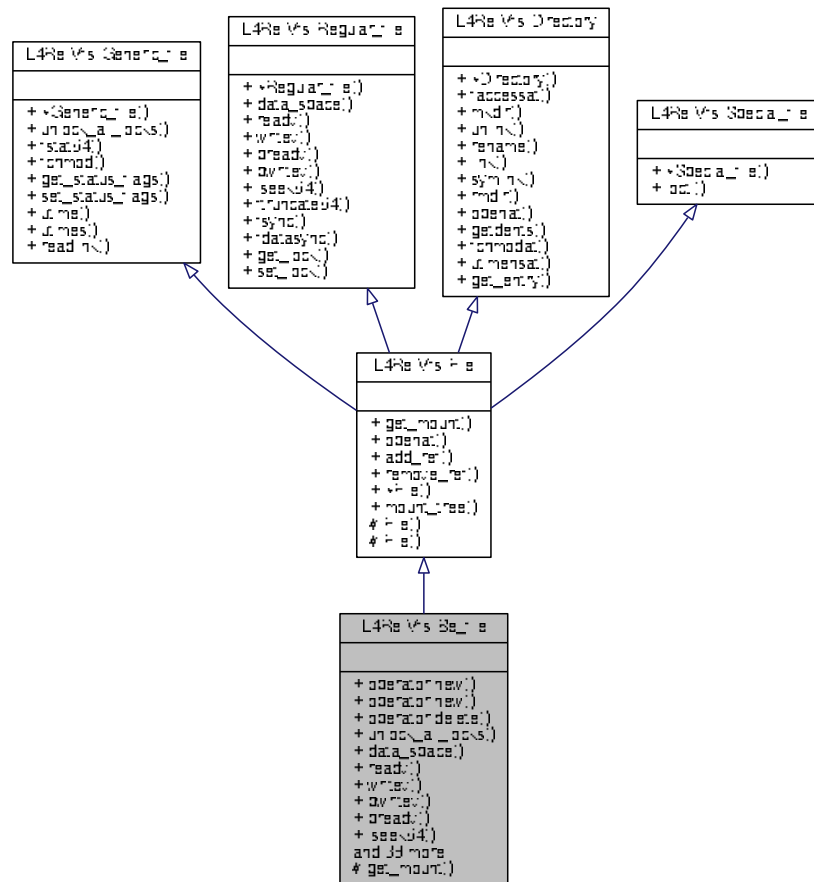
## 14.271 L4Re::Vfs::Be\_file Class Reference

Boiler plate class for implementing an open file for [L4Re::Vfs](#).

Inheritance diagram for L4Re::Vfs::Be\_file:



Collaboration diagram for L4Re::Vfs::Be\_file:



## Public Member Functions

- `int unlock_all_locks ()` throw ()  
Unlock all locks on the file.
- `L4::Cap< L4Re::Dataspace > data_space ()` const throw ()  
Get an *L4Re::Dataspace* object for the file.
- `ssize_t readv (const struct iovec *, int)` throw ()  
Default backend for POSIX read and readv functions.
- `ssize_t writev (const struct iovec *, int)` throw ()  
Default backend for POSIX write and writev functions.
- `ssize_t pwritev (const struct iovec *, int, off64_t)` throw ()  
Default backend for POSIX pwrite and pwritev functions.
- `ssize_t preadv (const struct iovec *, int, off64_t)` throw ()  
Default backend for POSIX pread and preadv functions.
- `off64_t lseek64 (off64_t, int)` throw ()  
Default backend for POSIX seek and lseek functions.
- `int truncate64 (off64_t)` throw ()  
Default backend for the POSIX truncate, ftruncate and similar functions.
- `int fsync ()` const throw ()

- Default backend for POSIX fsync.*

  - int [fdatasync](#) () const throw ()

*Default backend for POSIX fdatasync.*
- int [ioctl](#) (unsigned long, va\_list) throw ()

*Default backend for POSIX ioctl.*
- int [fstat64](#) (struct stat64 \*) const throw ()

*Get status information for the file.*
- int [fchmod](#) (mode\_t) throw ()

*Default backend for POSIX chmod and fchmod.*
- int [get\\_status\\_flags](#) () const throw ()

*Default backend for POSIX fcntl subfunctions.*
- int [set\\_status\\_flags](#) (long) throw ()

*Default backend for POSIX fcntl subfunctions.*
- int [get\\_lock](#) (struct flock64 \*) throw ()

*Default backend for POSIX fcntl subfunctions.*
- int [set\\_lock](#) (struct flock64 \*, bool) throw ()

*Default backend for POSIX fcntl subfunctions.*
- int [faccessat](#) (const char \*, int, int) throw ()

*Default backend for POSIX access and faccessat functions.*
- int [fchmodat](#) (const char \*, mode\_t, int) throw ()

*Default backend for POSIX fchmodat function.*
- int [utime](#) (const struct utimbuf \*) throw ()

*Default backend for POSIX utime.*
- int [utimes](#) (const struct timeval [2]) throw ()

*Default backend for POSIX utimes.*
- int [utimensat](#) (const char \*, const struct timespec [2], int) throw ()

*Default backend for POSIX utimensat.*
- int [mkdir](#) (const char \*, mode\_t) throw ()

*Default backend for POSIX mkdir and mkdirat.*
- int [unlink](#) (const char \*) throw ()

*Default backend for POSIX unlink, unlinkat.*
- int [rename](#) (const char \*, const char \*) throw ()

*Default backend for POSIX rename, renameat.*
- int [link](#) (const char \*, const char \*) throw ()

*Default backend for POSIX link, linkat.*
- int [symlink](#) (const char \*, const char \*) throw ()

*Default backend for POSIX symlink, symlinkat.*
- int [rmdir](#) (const char \*) throw ()

*Default backend for POSIX rmdir, rmdirat.*
- ssize\_t [readlink](#) (char \*, size\_t)

*Default backend for POSIX readlink, readlinkat.*

### 14.271.1 Detailed Description

Boiler plate class for implementing an open file for [L4Re::Vfs](#).

This class may be used as a base class for everything that a POSIX file descriptor may point to. This are things such as regular files, directories, special device files, streams, pipes, and so on.

Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 40 of file [backend](#).

## 14.271.2 Member Function Documentation

### 14.271.2.1 data\_space()

```
L4::Cap<L4Re::Dataspace> L4Re::Vfs::Be_file::data_space () const throw () [inline], [virtual]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

#### Note

mmap is not possible if the function returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#), that represents the file contents in an [L4Re](#) way.

Implements [L4Re::Vfs::Regular\\_file](#).

Definition at line 57 of file [backend](#).

### 14.271.2.2 fstat64()

```
int L4Re::Vfs::Be_file::fstat64 (
 struct stat64 * buf) const throw () [inline], [virtual]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

#### Parameters

|     |     |                                                    |
|-----|-----|----------------------------------------------------|
| out | buf | This buffer is filled with the status information. |
|-----|-----|----------------------------------------------------|

#### Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 96 of file [backend](#).

### 14.271.2.3 unlock\_all\_locks()

```
int L4Re::Vfs::Be_file::unlock_all_locks () throw () [inline], [virtual]
```

Unlock all locks on the file.

#### Note

All locks means all locks independent by which file the locks were taken.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

#### Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 53 of file [backend](#).

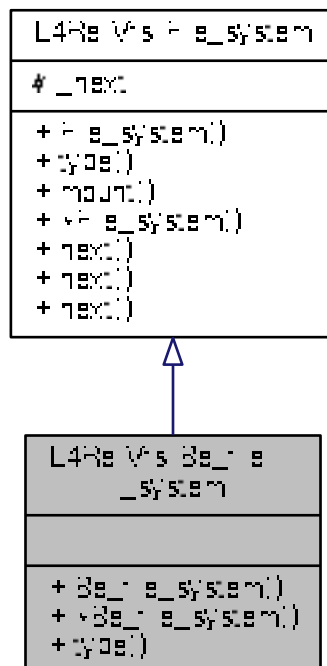
The documentation for this class was generated from the following file:

- l4/l4re\_vfs/backend

## 14.272 L4Re::Vfs::Be\_file\_system Class Reference

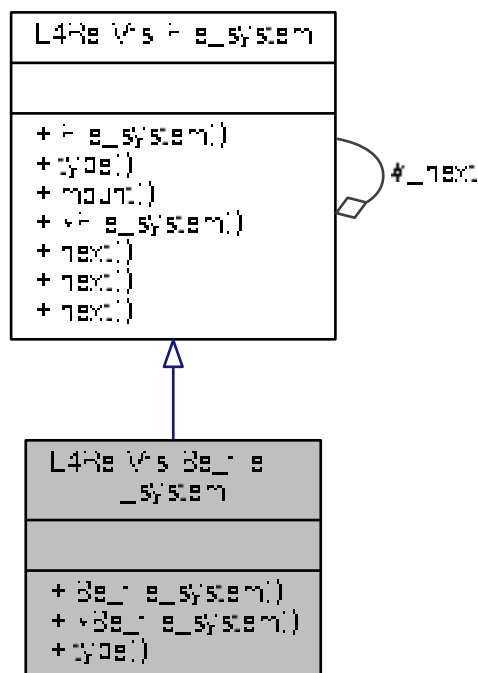
Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

Inheritance diagram for L4Re::Vfs::Be\_file\_system:





Collaboration diagram for L4Re::Vfs::Be\_file\_system:



## Public Member Functions

- [Be\\_file\\_system](#) (char const \*fstype) throw ()  
*Create a file-system object for the given fstype.*
- [~Be\\_file\\_system](#) () throw ()  
*Destroy a file-system object.*
- char const \* [type](#) () const throw ()  
*Return the file-system type.*

### 14.272.1 Detailed Description

Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

This class already takes care of registering and unregistering the file system in the global registry and implements the [type\(\)](#) method.

Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 308 of file [backend](#).

## 14.272.2 Constructor & Destructor Documentation

### 14.272.2.1 Be\_file\_system()

```
L4Re::Vfs::Be_file_system::Be_file_system (
 char const * fstype) throw () [inline], [explicit]
```

Create a file-system object for the given *fstype*.

#### Parameters

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>fstype</i> | The type that <a href="#">type()</a> shall return. |
|---------------|----------------------------------------------------|

This constructor takes care of registering the file system in the registry of L4Re::Vfs::vfs\_ops.

Definition at line 322 of file [backend](#).

### 14.272.2.2 ~Be\_file\_system()

```
L4Re::Vfs::Be_file_system::~~Be_file_system () throw () [inline]
```

Destroy a file-system object.

This destructor takes care of removing this file system from the registry of L4Re::Vfs::vfs\_ops.

Definition at line 334 of file [backend](#).

## 14.272.3 Member Function Documentation

### 14.272.3.1 type()

```
char const* L4Re::Vfs::Be_file_system::type () const throw () [inline], [virtual]
```

Return the file-system type.

Returns the file-system type given as *fstype* in the constructor.

Implements [L4Re::Vfs::File\\_system](#).

Definition at line 344 of file [backend](#).

The documentation for this class was generated from the following file:

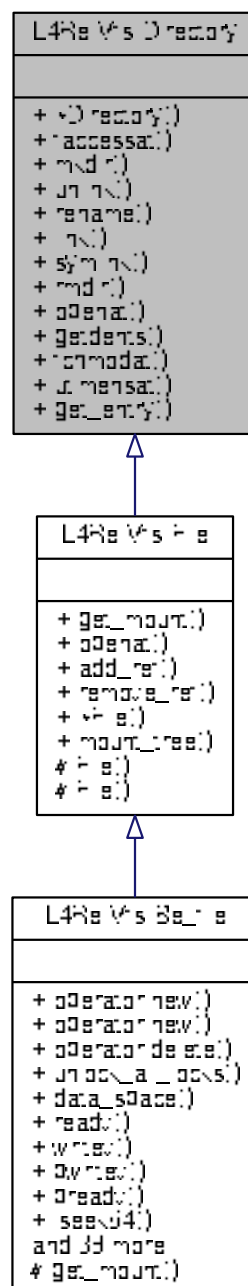
- [l4/l4re\\_vfs/backend](#)

## 14.273 L4Re::Vfs::Directory Class Reference

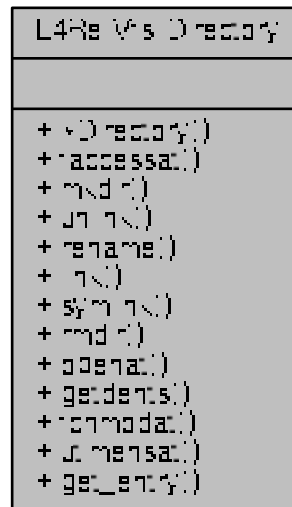
Interface for a POSIX file that is a directory.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:



Collaboration diagram for L4Re::Vfs::Directory:



## Public Member Functions

- virtual int [faccessat](#) (const char \*path, int mode, int flags)=0 throw ()  
*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode)=0 throw ()  
*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path)=0 throw ()  
*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*path)=0 throw ()  
*Delete an empty directory.*

### 14.273.1 Detailed Description

Interface for a POSIX file that is a directory.

This interface provides functionality for directory files in the [L4Re::Vfs](#). However, real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line 139 of file [vfs.h](#).

## 14.273.2 Member Function Documentation

### 14.273.2.1 faccessat()

```
virtual int L4Re::Vfs::Directory::faccessat (
 const char * path,
 int mode,
 int flags) throw () [pure virtual]
```

Check access permissions on the given file.

Backend function for POSIX access and faccessat functions.

#### Parameters

|              |                                                                                                                      |
|--------------|----------------------------------------------------------------------------------------------------------------------|
| <i>path</i>  | The path relative to this directory. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
| <i>mode</i>  | The access mode to check.                                                                                            |
| <i>flags</i> | The flags as in POSIX faccessat (AT_EACCESS, AT_SYMLINK_NOFOLLOW).                                                   |

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.273.2.2 link()

```
virtual int L4Re::Vfs::Directory::link (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

Create a hard link (second name) for the given file.

Backend for the POSIX link and linkat functions.

#### Parameters

|                 |                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>src_path</i> | The old name to the file. Note: <i>src_path</i> is relative to this directory and may contain subdirectories.           |
| <i>dst_path</i> | The new (second) name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories. |

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.273.2.3 mkdir()

```
virtual int L4Re::Vfs::Directory::mkdir (
 const char * path,
 mode_t mode) throw () [pure virtual]
```

Create a new subdirectory.

Backend for POSIX mkdir and mkdirat function calls.

#### Parameters

|             |                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>path</i> | The name of the subdirectory to create. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
| <i>mode</i> | The file mode to use for the new directory.                                                                             |

#### Returns

0 on success, or <0 on error. -ENOTDIR if this or some component in path is is not a directory.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.273.2.4 rename()

```
virtual int L4Re::Vfs::Directory::rename (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

Rename the given file.

Backend for the POSIX rename, renameat functions.

#### Parameters

|                 |                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>src_path</i> | The old name to the file to rename. Note: <i>src_path</i> is relative to this directory and may contain subdirectories. |
| <i>dst_path</i> | The new name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories.          |

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

## 14.273.2.5 rmdir()

```
virtual int L4Re::Vfs::Directory::rmdir (
 const char * path) throw () [pure virtual]
```

Delete an empty directory.

Backend for POSIX rmdir, rmdirat functions.

## Parameters

|             |                                                                                                                      |
|-------------|----------------------------------------------------------------------------------------------------------------------|
| <i>path</i> | The name of the directory to remove. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
|-------------|----------------------------------------------------------------------------------------------------------------------|

## Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

## 14.273.2.6 symlink()

```
virtual int L4Re::Vfs::Directory::symlink (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

## Parameters

|                 |                                                                                                           |
|-----------------|-----------------------------------------------------------------------------------------------------------|
| <i>src_path</i> | The old name to the file. Note: <i>src_path</i> shall be an absolute path.                                |
| <i>dst_path</i> | The name for symlink. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories. |

## Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

## 14.273.2.7 unlink()

```
virtual int L4Re::Vfs::Directory::unlink (
 const char * path) throw () [pure virtual]
```

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

## Parameters

|             |                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------|
| <i>path</i> | The name to the file to unlink. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
|-------------|-----------------------------------------------------------------------------------------------------------------|

## Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

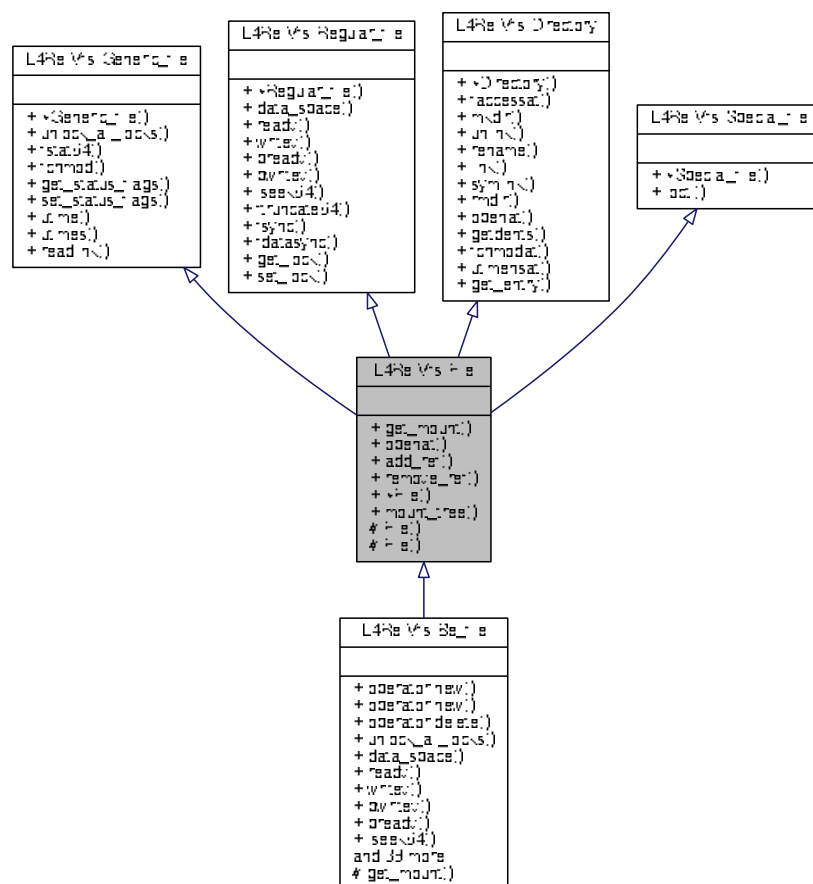
- `l4/l4re_vfs/vfs.h`

## 14.274 L4Re::Vfs::File Class Reference

The basic interface for an open POSIX file.

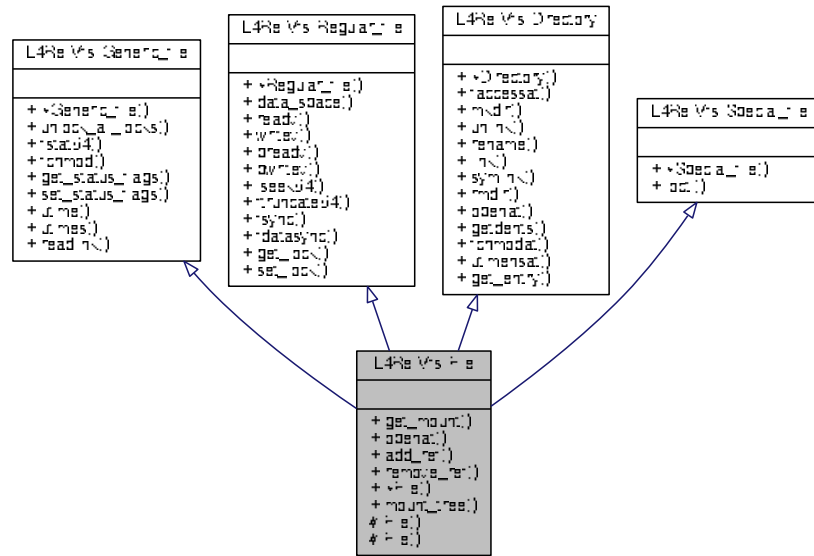
```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:





Collaboration diagram for L4Re::Vfs::File:



## Additional Inherited Members

### 14.274.1 Detailed Description

The basic interface for an open POSIX file.

An open POSIX file can be anything that hides behind a POSIX file descriptor. This means that even a directories are files. An open file can be anything from a directory to a special device file so see [Generic\\_file](#), [Regular\\_file](#), [Directory](#), and [Special\\_file](#) for more information.

#### Note

For implementing a backend for the [L4Re::Vfs](#) you may use [L4Re::Vfs::Be\\_file](#) as a base class.

Definition at line 434 of file [vfs.h](#).

The documentation for this class was generated from the following file:

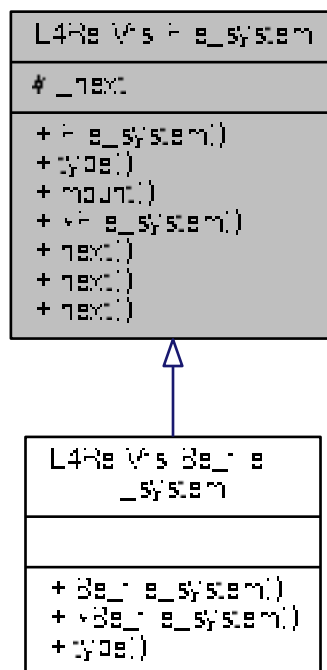
- [l4/l4re\\_vfs/vfs.h](#)

## 14.275 L4Re::Vfs::File\_system Class Reference

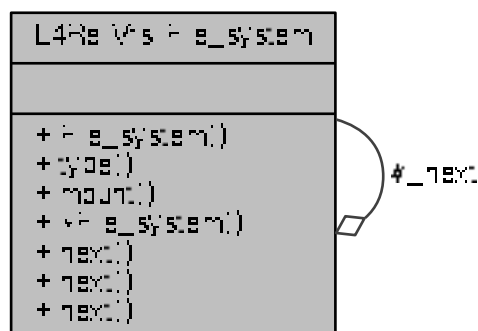
Basic interface for an [L4Re::Vfs](#) file system.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File\_system:



Collaboration diagram for L4Re::Vfs::File\_system:



## Public Member Functions

- virtual char const \* [type](#) () const =0 throw ()  
*Returns the type of the file system, used in mount as fstype argument.*
- virtual int [mount](#) (char const \*source, unsigned long mountflags, void const \*data, [cxx::Ref\\_ptr< File >](#) \*dir)=0 throw ()  
*Create a directory object dir representing source mounted with this file system.*

### 14.275.1 Detailed Description

Basic interface for an [L4Re::Vfs](#) file system.

#### Note

For implementing a special file system you may use [L4Re::Vfs::Be\\_file\\_system](#) as a base class.

The may purpose of this interface is that there is a single object for each supported file-system type (e.g., ext2, vfat) exists in your application and is registered at the [L4Re::Vfs::Fs](#) singleton available in via [L4Re::Vfs::vfs\\_ops](#). At the end the POSIX mount function call the [File\\_system::mount](#) method for the given file-system type given in mount.

Definition at line [821](#) of file [vfs.h](#).

### 14.275.2 Member Function Documentation

#### 14.275.2.1 mount()

```
virtual int L4Re::Vfs::File_system::mount (
 char const * source,
 unsigned long mountflags,
 void const * data,
 cxx::Ref_ptr< File > * dir) throw) [pure virtual]
```

Create a directory object *dir* representing *source* mounted with this file system.

#### Parameters

|                   |                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------|
| <i>source</i>     | The path to the source device to mount. This may also be some URL or anything file-system specific. |
| <i>mountflags</i> | The mount flags as specified in the POSIX mount call.                                               |
| <i>data</i>       | The data as specified in the POSIX mount call. The contents are file-system specific.               |

#### Return values

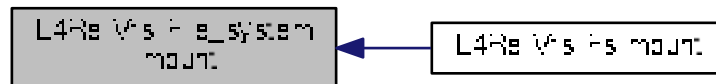
|            |                                                                     |
|------------|---------------------------------------------------------------------|
| <i>dir</i> | A new directory object representing the file-system root directory. |
|------------|---------------------------------------------------------------------|

**Returns**

0 on success, and <0 on error (e.g. -EINVAL).

Referenced by [L4Re::Vfs::Fs::mount\(\)](#).

Here is the caller graph for this function:

**14.275.2.2 type()**

```
virtual char const* L4Re::Vfs::File_system::type () const throw () [pure virtual]
```

Returns the type of the file system, used in mount as fstype argument.

**Note**

This method is already provided by [Be\\_file\\_system](#).

Implemented in [L4Re::Vfs::Be\\_file\\_system](#).

The documentation for this class was generated from the following file:

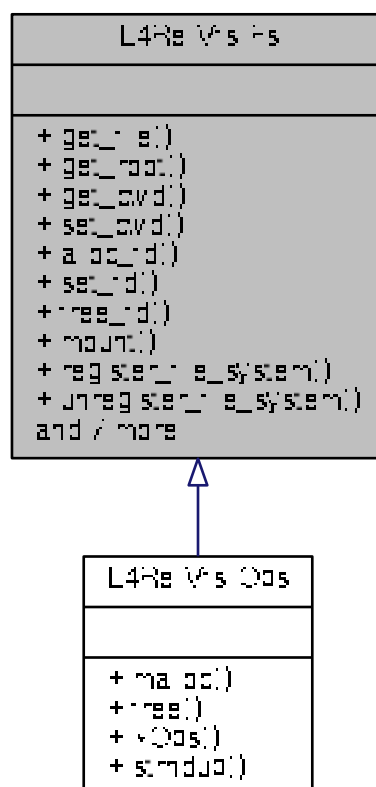
- l4/l4re\_vfs/vfs.h

**14.276 L4Re::Vfs::Fs Class Reference**

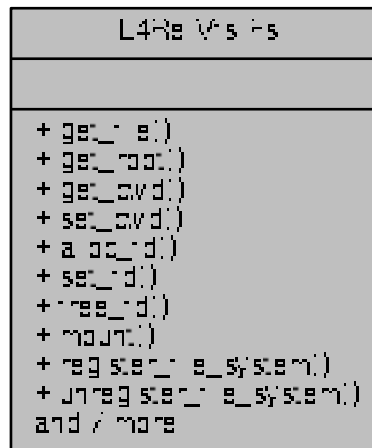
POSIX File-system related functionality.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Fs:



Collaboration diagram for L4Re::Vfs::Fs:



## Public Member Functions

- virtual [cxx::Ref\\_ptr< File > get\\_file](#) (int fd)=0 throw ()  
*Get the [L4Re::Vfs::File](#) for the file descriptor fd.*
- virtual [cxx::Ref\\_ptr< File > get\\_root](#) ()=0 throw ()  
*Get the directory object for the applications root directory.*
- virtual [cxx::Ref\\_ptr< File > get\\_cwd](#) () throw ()  
*Get the directory object for the applications current working directory.*
- virtual void [set\\_cwd](#) ([cxx::Ref\\_ptr< File > const &](#)) throw ()  
*Set the current working directory for the application.*
- virtual int [alloc\\_fd](#) ([cxx::Ref\\_ptr< File > const &](#)=[cxx::Ref\\_ptr<>::Nil](#))=0 throw ()  
*Allocate the next free file descriptor.*
- virtual [cxx::Ref\\_ptr< File > set\\_fd](#) (int fd, [cxx::Ref\\_ptr< File > const &](#)=[cxx::Ref\\_ptr<>::Nil](#))=0 throw ()  
*Set the file object referenced by the file descriptor fd.*
- virtual [cxx::Ref\\_ptr< File > free\\_fd](#) (int fd)=0 throw ()  
*Free the file descriptor fd.*
- virtual int [mount](#) (char const \*path, [cxx::Ref\\_ptr< File > const &](#)dir)=0 throw ()  
*Mount a given file object at the given global path in the VFS.*
- int [mount](#) (char const \*source, char const \*target, char const \*fstype, unsigned long mountflags, void const \*data) throw ()  
*Backend for the POSIX mount call.*

### 14.276.1 Detailed Description

POSIX File-system related functionality.

**Note**

This class usually exists as a singleton as a superclass of [L4Re::Vfs::Ops](#) (

**See also**

[L4Re::Vfs::vfs\\_ops](#)).

Definition at line [873](#) of file [vfs.h](#).

## 14.276.2 Member Function Documentation

### 14.276.2.1 `alloc_fd()`

```
virtual int L4Re::Vfs::Fs::alloc_fd (
 cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) throw () [pure virtual]
```

Allocate the next free file descriptor.

**Parameters**

|          |                                             |
|----------|---------------------------------------------|
| <i>f</i> | The file to assign to that file descriptor. |
|----------|---------------------------------------------|

**Returns**

the allocated file descriptor, or -EMFILE on error.

### 14.276.2.2 `free_fd()`

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::free_fd (
 int fd) throw () [pure virtual]
```

Free the file descriptor *fd*.

**Parameters**

|           |                              |
|-----------|------------------------------|
| <i>fd</i> | The file descriptor to free. |
|-----------|------------------------------|

**Returns**

A pointer to the file object that was assigned to the *fd*.

### 14.276.2.3 `get_file()`

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::get_file (
 int fd) throw () [pure virtual]
```

Get the [L4Re::Vfs::File](#) for the file descriptor *fd*.

#### Parameters

|           |                                   |
|-----------|-----------------------------------|
| <i>fd</i> | The POSIX file descriptor number. |
|-----------|-----------------------------------|

#### Returns

A pointer to the [File](#) object, or 0 if *fd* is not open.

### 14.276.2.4 `mount()`

```
virtual int L4Re::Vfs::Fs::mount (
 char const * path,
 cxx::Ref_ptr< File > const & dir) throw () [pure virtual]
```

Mount a given file object at the given global path in the VFS.

#### Parameters

|             |                                                                               |
|-------------|-------------------------------------------------------------------------------|
| <i>path</i> | The global path to mount <i>dir</i> at.                                       |
| <i>dir</i>  | A pointer to the file/directory object that shall be mounted at <i>path</i> . |

#### Returns

0 on success, or <0 on error.

### 14.276.2.5 `set_fd()`

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::set_fd (
 int fd,
 cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) throw () [pure virtual]
```

Set the file object referenced by the file descriptor *fd*.

#### Parameters

|           |                                          |
|-----------|------------------------------------------|
| <i>fd</i> | The file descriptor to set to <i>f</i> , |
| <i>f</i>  | The file object to assign.               |



**Returns**

A pointer to the file object that was previously assigned to fd.

The documentation for this class was generated from the following file:

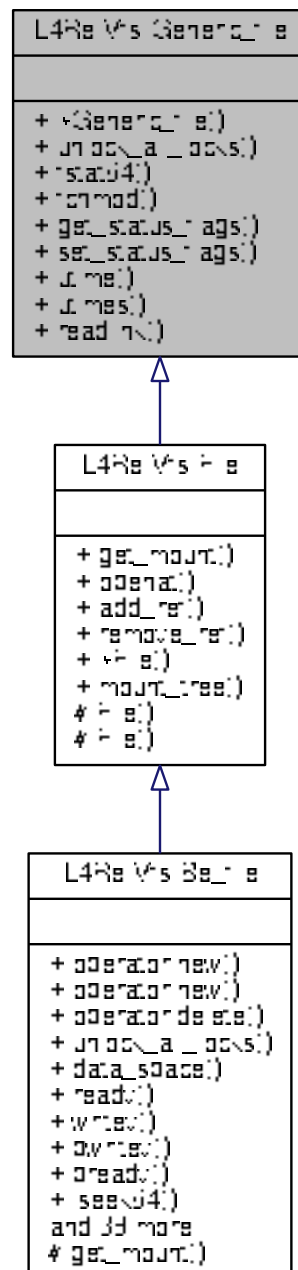
- l4/l4re\_vfs/vfs.h

## 14.277 L4Re::Vfs::Generic\_file Class Reference

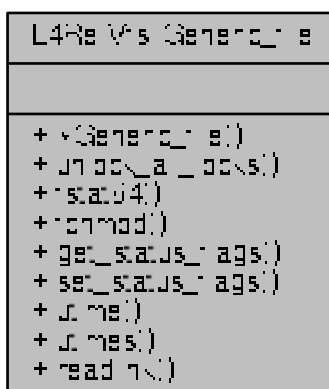
The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic\_file:



Collaboration diagram for L4Re::Vfs::Generic\_file:



## Public Member Functions

- virtual int [unlock\\_all\\_locks](#) ()=0 throw ()  
*Unlock all locks on the file.*
- virtual int [fstat64](#) (struct stat64 \*buf) const =0 throw ()  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t)=0 throw ()  
*Change POSIX access rights on that file.*
- virtual int [get\\_status\\_flags](#) () const =0 throw ()  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags)=0 throw ()  
*Set file status flags (fcntl F\_SETFL).*

### 14.277.1 Detailed Description

The common interface for an open POSIX file.

This interface is common to all kinds of open files, independent of the file type (e.g., directory, regular file etc.). However, in the [L4Re::Vfs](#) the interface [File](#) is used for every real object.

See also

[L4Re::Vfs::File](#) for mor information.

Definition at line 61 of file [vfs.h](#).

### 14.277.2 Member Function Documentation

### 14.277.2.1 fchmod()

```
virtual int L4Re::Vfs::Generic_file::fchmod (
 mode_t) throw () [pure virtual]
```

Change POSIX access rights on that file.

Backend for POSIX chmod and fchmod.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.277.2.2 fstat64()

```
virtual int L4Re::Vfs::Generic_file::fstat64 (
 struct stat64 * buf) const throw () [pure virtual]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

#### Parameters

|     |     |                                                    |
|-----|-----|----------------------------------------------------|
| out | buf | This buffer is filled with the status information. |
|-----|-----|----------------------------------------------------|

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.277.2.3 get\_status\_flags()

```
virtual int L4Re::Vfs::Generic_file::get_status_flags () const throw () [pure virtual]
```

Get file status flags (fcntl F\_GETFL).

This function is used by the fcntl implementation for the F\_GETFL command.

#### Returns

flags such as O\_RDONLY, O\_WRONLY, O\_RDWR, O\_DIRECT, O\_ASYNC, O\_NOATIME, O\_NONBLOCK, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.277.2.4 set\_status\_flags()

```
virtual int L4Re::Vfs::Generic_file::set_status_flags (
 long flags) throw () [pure virtual]
```

Set file status flags (fcntl F\_SETFL).

This function is used by the fcntl implementation for the F\_SETFL command.

**Parameters**

|              |                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <i>flags</i> | The file status flags to set. This must be a combination of O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_ASYNC, O_DIRECT, O_NOATIME, O_NONBLOCK. |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|

**Note**

Creation flags such as O\_CREAT, O\_EXCL, O\_NOCTTY, O\_TRUNC are ignored.

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**14.277.2.5 unlock\_all\_locks()**

```
virtual int L4Re::Vfs::Generic_file::unlock_all_locks () throw () [pure virtual]
```

Unlock all locks on the file.

**Note**

All locks means all locks independent by which file the locks were taken.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

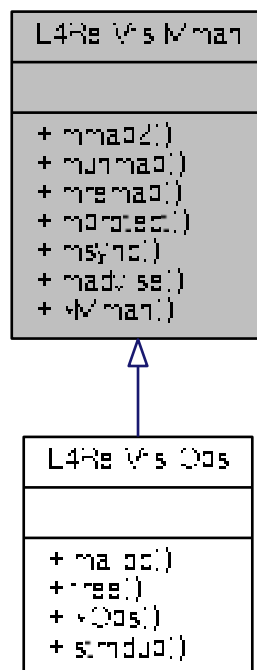
- l4/l4re\_vfs/vfs.h

## 14.278 L4Re::Vfs::Mman Class Reference

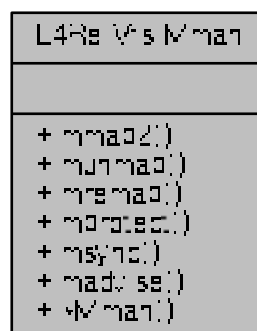
Interface for the POSIX memory management.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Mman:



Collaboration diagram for L4Re::Vfs::Mman:



## Public Member Functions

- virtual int [mmap2](#) (void \*start, size\_t len, int prot, int flags, int fd, off\_t offset, void \*\*ptr)=0 throw ()  
*Backend for the mmap2 system call.*
- virtual int [munmap](#) (void \*start, size\_t len)=0 throw ()  
*Backend for the munmap system call.*
- virtual int [mremap](#) (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_addr)=0 throw ()  
*Backend for the mremap system call.*
- virtual int [mprotect](#) (const void \*a, size\_t sz, int prot)=0 throw ()  
*Backend for the mprotect system call.*
- virtual int [msync](#) (void \*addr, size\_t len, int flags)=0 throw ()  
*Backend for the msync system call.*
- virtual int [madvice](#) (void \*addr, size\_t len, int advice)=0 throw ()  
*Backend for the madvice system call.*

### 14.278.1 Detailed Description

Interface for the POSIX memory management.

#### Note

This interface exists usually as a singleton as superclass of [L4Re::Vfs::Ops](#).

An implementation for this interface is in [l4/l4re\\_vfs/impl/vfs\\_impl.h](#) and used by the l4re\_vfs library or by the VFS implementation in ldso.

Definition at line [744](#) of file [vfs.h](#).

The documentation for this class was generated from the following file:

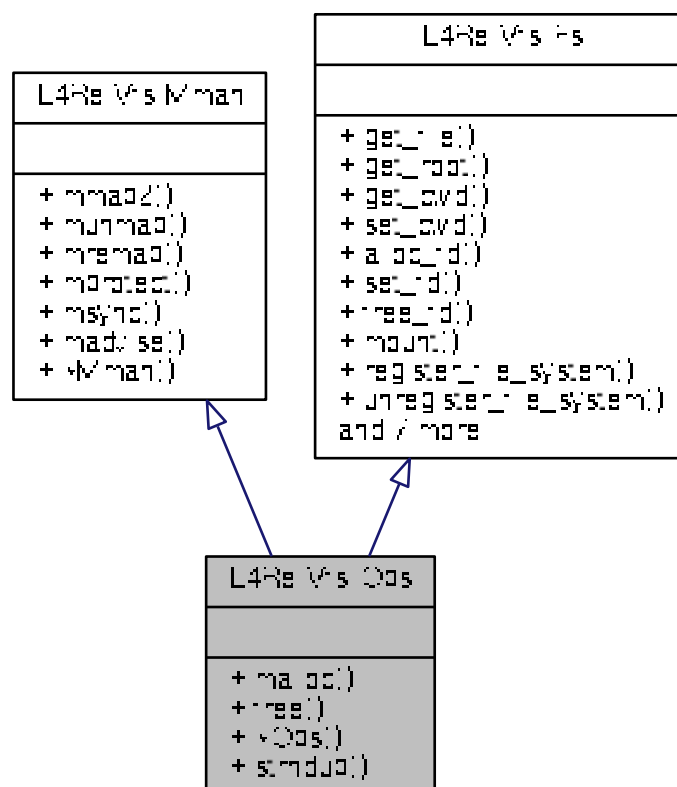
- [l4/l4re\\_vfs/vfs.h](#)

## 14.279 L4Re::Vfs::Ops Class Reference

Interface for the POSIX backends for an application.

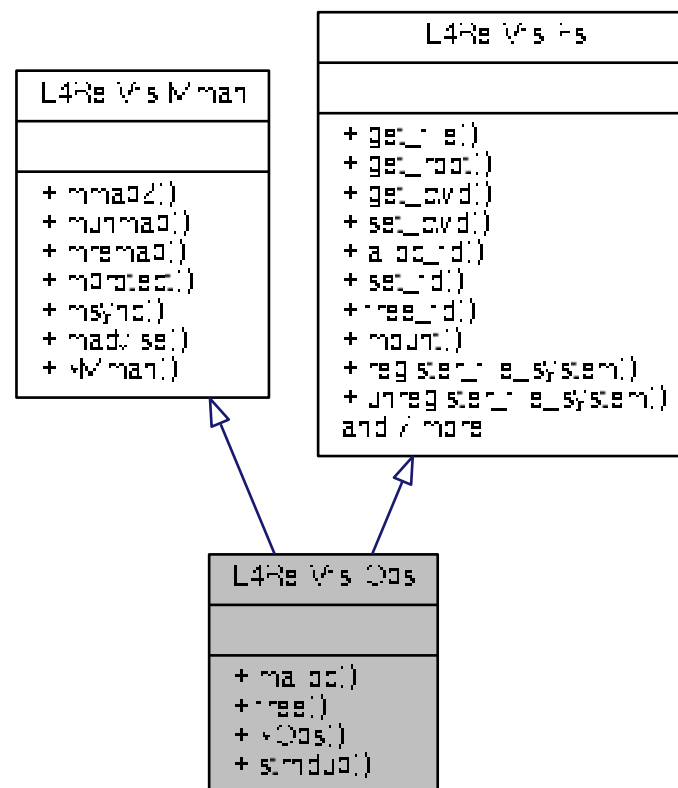
```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:





Collaboration diagram for L4Re::Vfs::Ops:



## Additional Inherited Members

### 14.279.1 Detailed Description

Interface for the POSIX backends for an application.

#### Note

There usually exists a single instance of this interface available via `L4Re::Vfs::vfs_ops` that is used for all kinds of C-Library functions.

Definition at line 994 of file [vfs.h](#).

The documentation for this class was generated from the following file:

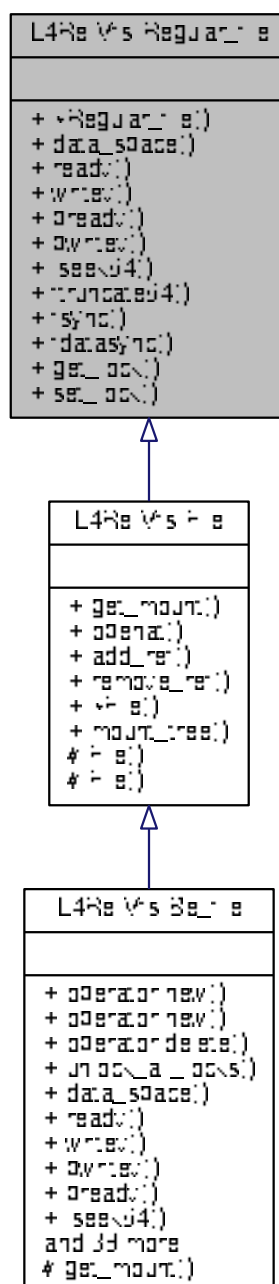
- `l4/l4re_vfs/vfs.h`

## 14.280 L4Re::Vfs::Regular\_file Class Reference

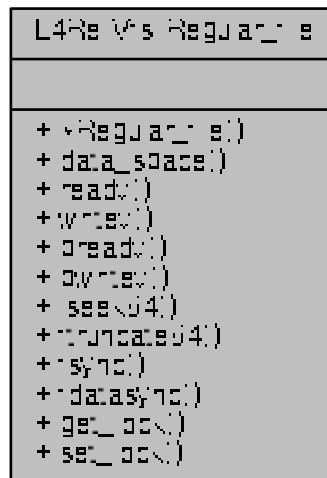
Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Regular\_file:



Collaboration diagram for L4Re::Vfs::Regular\_file:



## Public Member Functions

- virtual [L4::Cap< L4Re::Dataspace > data\\_space](#) () const =0 throw ()  
*Get an [L4Re::Dataspace](#) object for the file.*
- virtual ssize\_t [read](#) (const struct iovec \*, int iovcnt)=0 throw ()  
*Read one or more blocks of data from the file.*
- virtual ssize\_t [write](#) (const struct iovec \*, int iovcnt)=0 throw ()  
*Write one or more blocks of data to the file.*
- virtual off64\_t [lseek64](#) (off64\_t, int)=0 throw ()  
*Change the file pointer.*
- virtual int [ftruncate64](#) (off64\_t pos)=0 throw ()  
*Truncate the file at the given position.*
- virtual int [fsync](#) () const =0 throw ()  
*Sync the data and meta data to persistent storage.*
- virtual int [datasync](#) () const =0 throw ()  
*Sync the data to persistent storage.*
- virtual int [get\\_lock](#) (struct flock64 \*lock)=0 throw ()  
*Test if the given lock can be placed in the file.*
- virtual int [set\\_lock](#) (struct flock64 \*lock, bool wait)=0 throw ()  
*Acquire or release the given lock on the file.*

### 14.280.1 Detailed Description

Interface for a POSIX file that provides regular file semantics.

Real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line 266 of file [vfs.h](#).

## 14.280.2 Member Function Documentation

### 14.280.2.1 data\_space()

```
virtual L4::Cap<L4Re::Dataspace> L4Re::Vfs::Regular_file::data_space () const throw () [pure virtual]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

#### Note

mmap is not possible if the function returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#), that represents the file contents in an [L4Re](#) way.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.280.2.2 fdatsync()

```
virtual int L4Re::Vfs::Regular_file::fdatsync () const throw () [pure virtual]
```

Sync the data to persistent storage.

This is the backend for POSIX fdatsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.280.2.3 fsync()

```
virtual int L4Re::Vfs::Regular_file::fsync () const throw () [pure virtual]
```

Sync the data and meta data to persistent storage.

This is the backend for POSIX fsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.280.2.4 ftruncate64()

```
virtual int L4Re::Vfs::Regular_file::ftruncate64 (
 off64_t pos) throw () [pure virtual]
```

Truncate the file at the given position.

This function is the backend for truncate and friends.

**Parameters**

|            |                                                  |
|------------|--------------------------------------------------|
| <i>pos</i> | The offset at which the file shall be truncated. |
|------------|--------------------------------------------------|

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**14.280.2.5 get\_lock()**

```
virtual int L4Re::Vfs::Regular_file::get_lock (
 struct flock64 * lock) throw () [pure virtual]
```

Test if the given lock can be placed in the file.

This function is used as backend for fcntl F\_GETLK commands.

**Parameters**

|             |                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>lock</i> | The lock that shall be placed on the file. The <i>l_type</i> member will contain F_UNLCK if the lock could be placed. |
|-------------|-----------------------------------------------------------------------------------------------------------------------|

**Returns**

0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**14.280.2.6 lseek64()**

```
virtual off64_t L4Re::Vfs::Regular_file::lseek64 (
 off64_t ,
 int) throw () [pure virtual]
```

Change the file pointer.

This is the backend for POSIX seek, lseek and friends.

**Returns**

The new file position, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

#### 14.280.2.7 readv()

```
virtual ssize_t L4Re::Vfs::Regular_file::readv (
 const struct iovec * ,
 int iovcnt) throw () [pure virtual]
```

Read one or more blocks of data from the file.

This function acts as backend for POSIX read and readv calls and reads data starting for the `f_pos` pointer of that open file. The file pointer is advanced according to the number of read bytes.

##### Returns

The number of bytes read from the file. or <0 on error-

Implemented in [L4Re::Vfs::Be\\_file](#).

#### 14.280.2.8 set\_lock()

```
virtual int L4Re::Vfs::Regular_file::set_lock (
 struct flock64 * lock,
 bool wait) throw () [pure virtual]
```

Acquire or release the given lock on the file.

This function is used as backend for `fcntl F_SETLK` and `F_SETLKW` commands.

##### Parameters

|             |                                                                 |
|-------------|-----------------------------------------------------------------|
| <i>lock</i> | The lock that shall be placed on the file.                      |
| <i>wait</i> | If true, then block if there is a conflicting lock on the file. |

##### Returns

0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

#### 14.280.2.9 writev()

```
virtual ssize_t L4Re::Vfs::Regular_file::writev (
 const struct iovec * ,
 int iovcnt) throw () [pure virtual]
```

Write one or more blocks of data to the file.

This function acts as backend for POSIX write and writev calls. The data is written starting at the current file pointer and the file pointer must be advanced according to the number of written bytes.

#### Returns

The number of bytes written to the file, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

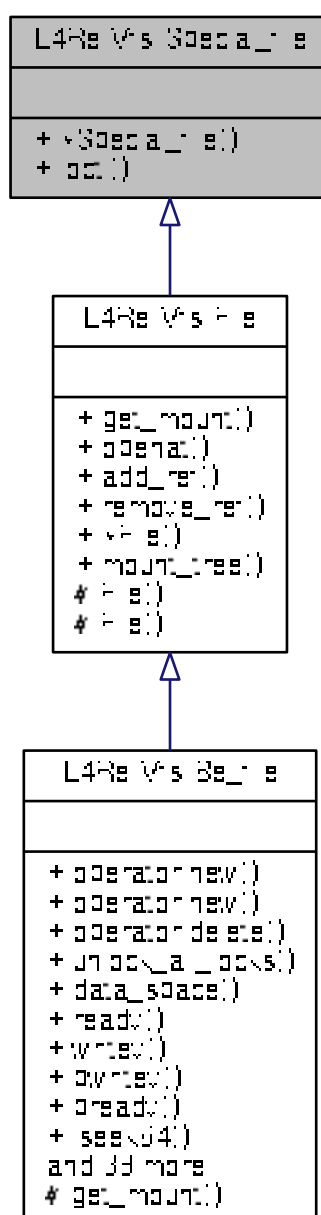
- l4/l4re\_vfs/vfs.h

## 14.281 L4Re::Vfs::Special\_file Class Reference

Interface for a POSIX file that provides special file semantics.

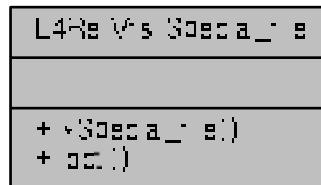
```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Special\_file:





Collaboration diagram for L4Re::Vfs::Special\_file:



## Public Member Functions

- virtual int [ioctl](#) (unsigned long cmd, va\_list args)=0 throw ()  
*The famous IO control.*

### 14.281.1 Detailed Description

Interface for a POSIX file that provides special file semantics.

Real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line [399](#) of file [vfs.h](#).

### 14.281.2 Member Function Documentation

#### 14.281.2.1 ioctl()

```
virtual int L4Re::Vfs::Special_file::ioctl (
 unsigned long cmd,
 va_list args) throw () [pure virtual]
```

The famous IO control.

Backend for POSIX generic object invocation ioctl.

#### Parameters

|             |                                                            |
|-------------|------------------------------------------------------------|
| <i>cmd</i>  | The ioctl command.                                         |
| <i>args</i> | The arguments for the ioctl, usually some kind of pointer. |

**Returns**

$\geq 0$  on success, or  $< 0$  on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

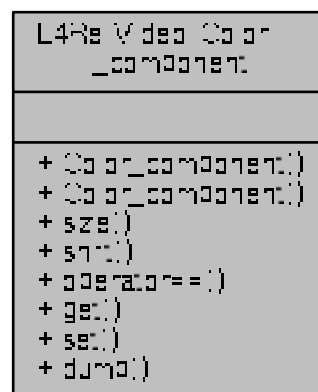
The documentation for this class was generated from the following file:

- [l4/l4re\\_vfs/vfs.h](#)

## 14.282 L4Re::Video::Color\_component Class Reference

A color component.

Collaboration diagram for L4Re::Video::Color\_component:



### Public Member Functions

- [Color\\_component](#) ()  
*Constructor.*
- [Color\\_component](#) (unsigned char bits, unsigned char [shift](#))  
*Constructor.*
- unsigned char [size](#) () const  
*Return the number of bits used by the component.*
- unsigned char [shift](#) () const  
*Return the position of the component in the pixel.*
- bool [operator==](#) ([Color\\_component](#) const &o) const  
*Compare for equality.*
- int [get](#) (unsigned long v) const  
*Get component from value (normalized to 16bits).*
- long unsigned [set](#) (int v) const  
*Transform 16bit normalized value to the component in the color space.*
- template<typename OUT >  
void [dump](#) (OUT &s) const  
*Dump information on the view information to a stream.*

### 14.282.1 Detailed Description

A color component.

Definition at line 31 of file [colors](#).

### 14.282.2 Constructor & Destructor Documentation

#### 14.282.2.1 Color\_component()

```
L4Re::Video::Color_component::Color_component (
 unsigned char bits,
 unsigned char shift) [inline]
```

Constructor.

##### Parameters

|              |                                                |
|--------------|------------------------------------------------|
| <i>bits</i>  | Number of bits used by the component           |
| <i>shift</i> | Position in bits of the component in the pixel |

Definition at line 46 of file [colors](#).

### 14.282.3 Member Function Documentation

#### 14.282.3.1 dump()

```
template<typename OUT >
void L4Re::Video::Color_component::dump (
 OUT & s) const [inline]
```

Dump information on the view information to a stream.

##### Parameters

|          |        |
|----------|--------|
| <i>s</i> | Stream |
|----------|--------|

##### Returns

The stream

Definition at line 93 of file [colors](#).

### 14.282.3.2 get()

```
int L4Re::Video::Color_component::get (
 unsigned long v) const [inline]
```

Get component from value (normalized to 16bits).

#### Parameters

|   |       |
|---|-------|
| v | Value |
|---|-------|

#### Returns

Converted value

Definition at line 73 of file [colors](#).

### 14.282.3.3 operator==( )

```
bool L4Re::Video::Color_component::operator== (
 Color_component const & o) const [inline]
```

Compare for equality.

#### Returns

True if the same components are described, false if not.

Definition at line 65 of file [colors](#).

### 14.282.3.4 set()

```
long unsigned L4Re::Video::Color_component::set (
 int v) const [inline]
```

Transform 16bit normalized value to the component in the color space.

#### Parameters

|   |                               |
|---|-------------------------------|
| v | Value return Converted value. |
|---|-------------------------------|

Definition at line 84 of file [colors](#).

## 14.282.3.5 shift()

```
unsigned char L4Re::Video::Color_component::shift () const [inline]
```

Return the position of the component in the pixel.

**Returns**

Position in bits of the component in the pixel

Definition at line 59 of file [colors](#).

## 14.282.3.6 size()

```
unsigned char L4Re::Video::Color_component::size () const [inline]
```

Return the number of bits used by the component.

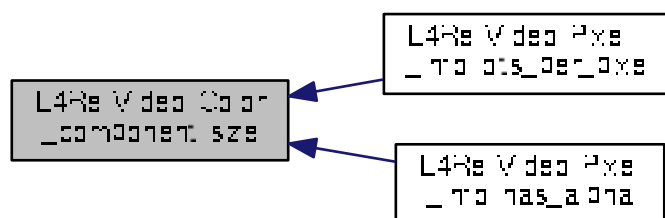
**Returns**

Number of bits used by the component

Definition at line 53 of file [colors](#).

Referenced by [L4Re::Video::Pixel\\_info::bits\\_per\\_pixel\(\)](#), and [L4Re::Video::Pixel\\_info::has\\_alpha\(\)](#).

Here is the caller graph for this function:



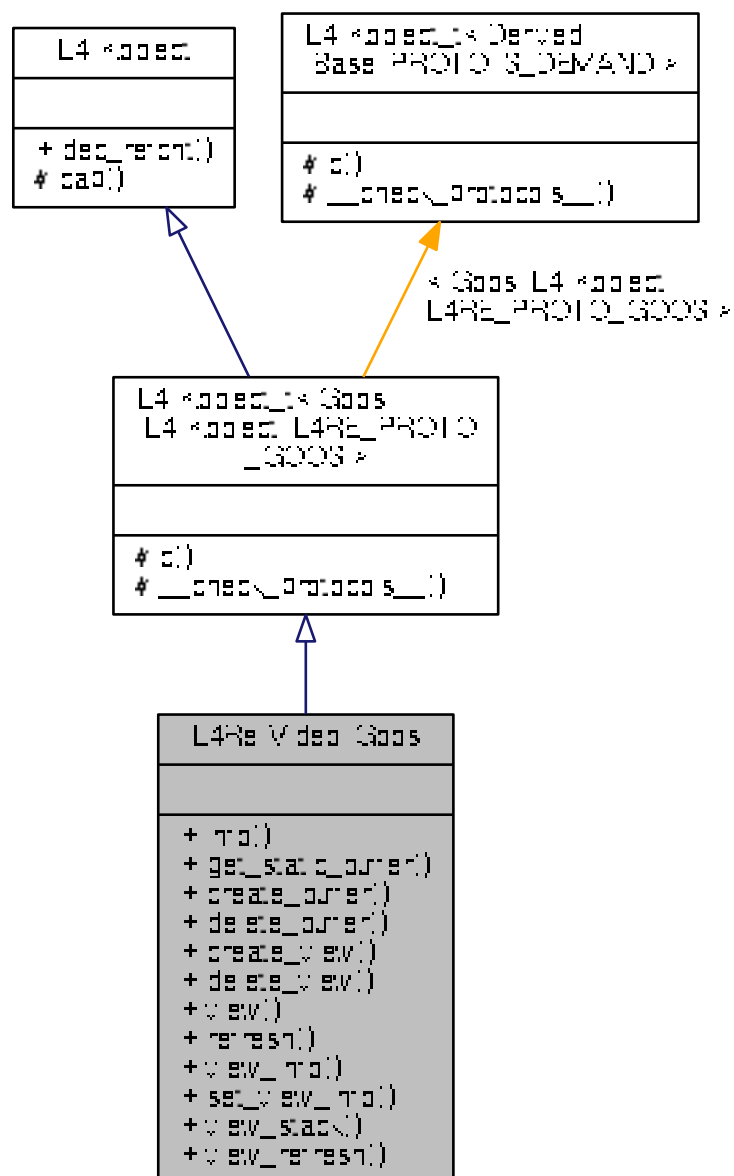
The documentation for this class was generated from the following file:

- `l4/re/video/colors`

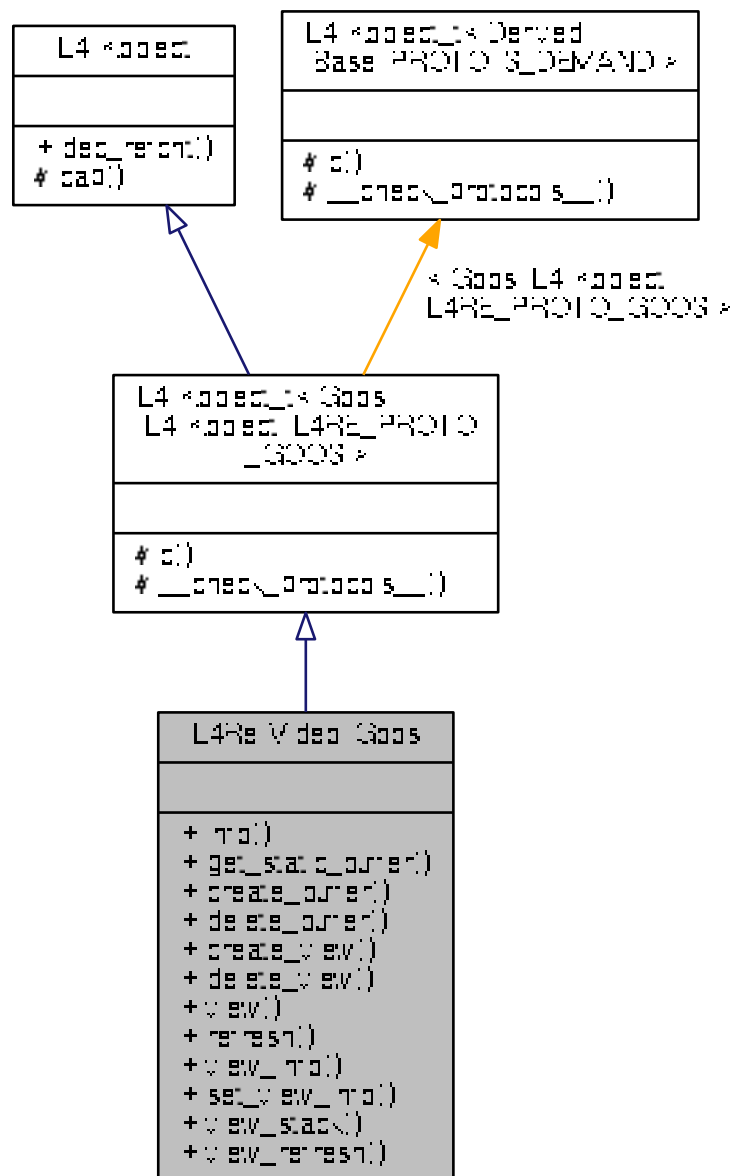
## 14.283 L4Re::Video::Goos Class Reference

A goos.

Inheritance diagram for L4Re::Video::Goos:



Collaboration diagram for L4Re::Video::Goos:



## Data Structures

- struct [Info](#)

*Information structure of a goos.*

## Public Types

- enum [Flags](#) { [F\\_auto\\_refresh](#) = 0x01, [F\\_pointer](#) = 0x02, [F\\_dynamic\\_views](#) = 0x04, [F\\_dynamic\\_buffers](#) = 0x08 }

*Flags for a goos.*

## Public Member Functions

- long [info](#) ([Info](#) \*info)  
*Return the goos information of the goos.*
- long [get\\_static\\_buffer](#) (unsigned idx, [L4::lpc::Out](#)< [L4::Cap](#)< [L4Re::Dataspace](#) > > rbuf)  
*Return a static buffer of a goos.*
- long [create\\_buffer](#) (unsigned long size, [L4::lpc::Out](#)< [L4::Cap](#)< [L4Re::Dataspace](#) > > rbuf)  
*Create a buffer.*
- long [delete\\_buffer](#) (unsigned idx)  
*Delete a buffer.*
- int [create\\_view](#) ([View](#) \*view, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) const throw ()  
*Create a view.*
- int [delete\\_view](#) ([View](#) const &v, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) const throw ()  
*Delete a view.*
- [View](#) [view](#) (unsigned index) const throw ()  
*Return a view.*
- long [refresh](#) (int x, int y, int w, int h)  
*Trigger refreshing of the given area on the virtual screen.*

## Additional Inherited Members

### 14.283.1 Detailed Description

A goos.

Definition at line 207 of file [goos](#).

### 14.283.2 Member Enumeration Documentation

#### 14.283.2.1 Flags

```
enum L4Re::Video::Goos::Flags
```

Flags for a goos.

#### Enumerator

|                                   |                                                  |
|-----------------------------------|--------------------------------------------------|
| <a href="#">F_auto_refresh</a>    | The graphics display is automatically refreshed. |
| <a href="#">F_pointer</a>         | We have a mouse pointer.                         |
| <a href="#">F_dynamic_views</a>   | Supports dynamically allocated views.            |
| <a href="#">F_dynamic_buffers</a> | Supports dynamically allocated buffers.          |

Definition at line 212 of file [goos](#).



### 14.283.3 Member Function Documentation

#### 14.283.3.1 create\_buffer()

```
long L4Re::Video::Goos::create_buffer (
 unsigned long size,
 L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)
```

Create a buffer.

##### Parameters

|             |                                                   |
|-------------|---------------------------------------------------|
| <i>size</i> | Size of buffer in bytes.                          |
| <i>rbuf</i> | Capability slot to point the buffer dataspace to. |

##### Return values

|          |                                                  |
|----------|--------------------------------------------------|
| $\geq 0$ | Success, the value returned is the buffer index. |
| $< 0$    | Error                                            |

#### 14.283.3.2 create\_view()

```
int L4Re::Video::Goos::create_view (
 View * view,
 l4_utcb_t * utcb = l4_utcb()) const throw() [inline]
```

Create a view.

##### Parameters

|     |             |                                                  |
|-----|-------------|--------------------------------------------------|
| out | <i>view</i> | A view object.                                   |
|     | <i>utcb</i> | UTCB of the caller. This is a default parameter. |

##### Return values

|          |                                                |
|----------|------------------------------------------------|
| $\geq 0$ | Success, the value returned is the view index. |
| $< 0$    | Error                                          |

Definition at line 296 of file [goos](#).

References [L4\\_INLINE\\_RPC\\_NF](#).

**14.283.3.3 delete\_buffer()**

```
long L4Re::Video::Goos::delete_buffer (
 unsigned idx)
```

Delete a buffer.

**Parameters**

|            |                   |
|------------|-------------------|
| <i>idx</i> | Buffer to delete. |
|------------|-------------------|

**Return values**

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

**14.283.3.4 delete\_view()**

```
int L4Re::Video::Goos::delete_view (
 View const & v,
 l4_utcb_t * utcb = l4_utcb()) const throw () [inline]
```

Delete a view.

**Parameters**

|             |                                                  |
|-------------|--------------------------------------------------|
| <i>v</i>    | The view object to delete.                       |
| <i>utcb</i> | UTCB of the caller. This is a default parameter. |

**Return values**

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 316 of file [goos](#).

References [L4\\_INLINE\\_RPC](#).

**14.283.3.5 get\_static\_buffer()**

```
long L4Re::Video::Goos::get_static_buffer (
 unsigned idx,
 L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)
```

Return a static buffer of a goos.

## Parameters

|             |                                                   |
|-------------|---------------------------------------------------|
| <i>idx</i>  | Index of the static buffer.                       |
| <i>rbuf</i> | Capability slot to point the buffer dataspace to. |

## Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

## 14.283.3.6 info()

```
long L4Re::Video::Goos::info (
 Info * info)
```

Return the goos information of the goos.

## Parameters

|     |             |                                                     |
|-----|-------------|-----------------------------------------------------|
| out | <i>info</i> | <a href="#">Goos</a> information structure pointer. |
|-----|-------------|-----------------------------------------------------|

## Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

## 14.283.3.7 view()

```
View L4Re::Video::Goos::view (
 unsigned index) const throw () [inline]
```

Return a view.

## Parameters

|              |                              |
|--------------|------------------------------|
| <i>index</i> | Index of the view to return. |
|--------------|------------------------------|

## Returns

The view.

Definition at line 347 of file [goos](#).

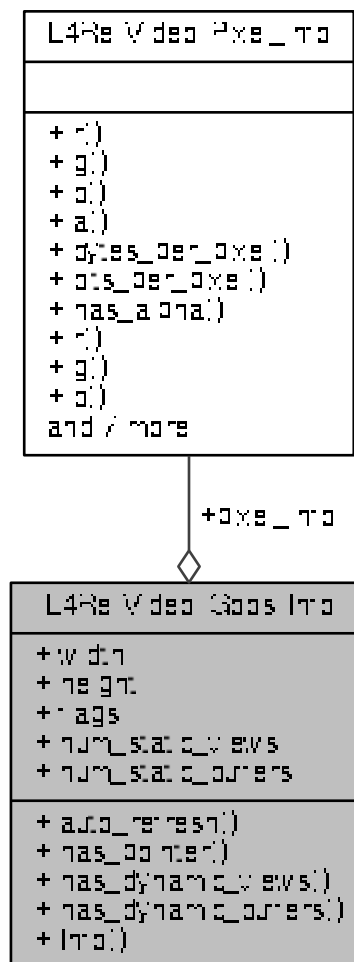
The documentation for this class was generated from the following file:

- l4/re/video/goos

## 14.284 L4Re::Video::Goos::Info Struct Reference

Information structure of a goos.

Collaboration diagram for L4Re::Video::Goos::Info:



### Public Member Functions

- bool [auto\\_refresh](#) () const  
*Return whether this goos does auto refreshing or the view refresh functions must be used to make changes visible.*
- bool [has\\_pointer](#) () const  
*Return whether a pointer is used by the provider of the goos.*
- bool [has\\_dynamic\\_views](#) () const  
*Return whether dynamic view are supported.*
- bool [has\\_dynamic\\_buffers](#) () const  
*Return whether dynamic buffers are supported.*

## Data Fields

- unsigned long [width](#)  
*Width.*
- unsigned long [height](#)  
*Height.*
- unsigned [flags](#)  
*Flags, see [Flags](#).*
- unsigned [num\\_static\\_views](#)  
*Number of static view.*
- unsigned [num\\_static\\_buffers](#)  
*Number of static buffers.*
- [Pixel\\_info](#) [pixel\\_info](#)  
*Pixel information.*

### 14.284.1 Detailed Description

Information structure of a goos.

Definition at line [221](#) of file [goos](#).

### 14.284.2 Member Function Documentation

#### 14.284.2.1 [auto\\_refresh\(\)](#)

```
bool L4Re::Video::Goos::Info::auto_refresh () const [inline]
```

Return whether this goos does auto refreshing or the view refresh functions must be used to make changes visible.

Definition at line [232](#) of file [goos](#).

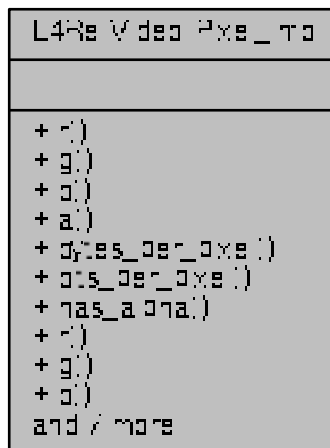
The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

## 14.285 L4Re::Video::Pixel\_info Class Reference

Pixel information.

Collaboration diagram for L4Re::Video::Pixel\_info:



### Public Member Functions

- [Color\\_component](#) const & [r](#) () const  
*Return the red color component of the pixel.*
- [Color\\_component](#) const & [g](#) () const  
*Return the green color component of the pixel.*
- [Color\\_component](#) const & [b](#) () const  
*Return the blue color component of the pixel.*
- [Color\\_component](#) const & [a](#) () const  
*Return the alpha color component of the pixel.*
- unsigned char [bytes\\_per\\_pixel](#) () const  
*Query size of pixel in bytes.*
- unsigned char [bits\\_per\\_pixel](#) () const  
*Number of bits of the pixel.*
- bool [has\\_alpha](#) () const  
*Return whether the pixel has an alpha channel.*
- void [r](#) ([Color\\_component](#) const &c)  
*Set the red color component of the pixel.*
- void [g](#) ([Color\\_component](#) const &c)  
*Set the green color component of the pixel.*
- void [b](#) ([Color\\_component](#) const &c)  
*Set the blue color component of the pixel.*
- void [a](#) ([Color\\_component](#) const &c)  
*Set the alpha color component of the pixel.*

- void [bytes\\_per\\_pixel](#) (unsigned char bpp)  
*Set the size of the pixel in bytes.*
- [Pixel\\_info](#) ()  
*Constructor.*
- [Pixel\\_info](#) (unsigned char bpp, char [r](#), char [rs](#), char [g](#), char [gs](#), char [b](#), char [bs](#), char [a](#)=0, char [as](#)=0)  
*Constructor.*
- template<typename VBI >  
[Pixel\\_info](#) (VBI const \*vbi)  
*Convenience constructor.*
- bool [operator==](#) ([Pixel\\_info](#) const &o) const  
*Compare for complete equality of the color space.*
- template<typename OUT >  
void [dump](#) (OUT &s) const  
*Dump information on the pixel to a stream.*

### 14.285.1 Detailed Description

Pixel information.

This class wraps the information on a pixel, such as the size and position of each color component in the pixel.

Definition at line [106](#) of file [colors](#).

### 14.285.2 Constructor & Destructor Documentation

#### 14.285.2.1 [Pixel\\_info\(\)](#) [1/2]

```
L4Re::Video::Pixel_info::Pixel_info (
 unsigned char bpp,
 char r,
 char rs,
 char g,
 char gs,
 char b,
 char bs,
 char a = 0,
 char as = 0) [inline]
```

Constructor.

#### Parameters

|            |                                       |
|------------|---------------------------------------|
| <i>bpp</i> | Size of pixel in bytes.               |
| <i>r</i>   | Red component size.                   |
| <i>rs</i>  | Red component shift.                  |
| <i>g</i>   | Green component size.                 |
| <i>gs</i>  | Green component shift.                |
| <i>b</i>   | Blue component size.                  |
| <i>bs</i>  | Blue component shift.                 |
| <i>a</i>   | Alpha component size, defaults to 0.  |
| <i>as</i>  | Alpha component shift, defaults to 0. |

Definition at line 203 of file [colors](#).

#### 14.285.2.2 Pixel\_info() [2/2]

```
template<typename VBI >
L4Re::Video::Pixel_info::Pixel_info (
 VBI const * vbi) [inline], [explicit]
```

Convenience constructor.

##### Parameters

|            |                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------|
| <i>vbi</i> | Suitable information structure. Convenience constructor to create the pixel info from a VESA Framebuffer Info. |
|------------|----------------------------------------------------------------------------------------------------------------|

Definition at line 215 of file [colors](#).

### 14.285.3 Member Function Documentation

#### 14.285.3.1 a() [1/2]

```
Color_component const& L4Re::Video::Pixel_info::a () const [inline]
```

Return the alpha color component of the pixel.

##### Returns

Alpha color component.

Definition at line 135 of file [colors](#).

#### 14.285.3.2 a() [2/2]

```
void L4Re::Video::Pixel_info::a (
 Color_component const & c) [inline]
```

Set the alpha color component of the pixel.

##### Parameters

|          |                        |
|----------|------------------------|
| <i>c</i> | Alpha color component. |
|----------|------------------------|



Definition at line 178 of file [colors](#).

#### 14.285.3.3 `b()` [1/2]

```
Color_component const& L4Re::Video::Pixel_info::b () const [inline]
```

Return the blue color component of the pixel.

##### Returns

Blue color component.

Definition at line 129 of file [colors](#).

#### 14.285.3.4 `b()` [2/2]

```
void L4Re::Video::Pixel_info::b (
 Color_component const & c) [inline]
```

Set the blue color component of the pixel.

##### Parameters

|                |                       |
|----------------|-----------------------|
| <code>c</code> | Blue color component. |
|----------------|-----------------------|

Definition at line 172 of file [colors](#).

#### 14.285.3.5 `bits_per_pixel()`

```
unsigned char L4Re::Video::Pixel_info::bits_per_pixel () const [inline]
```

Number of bits of the pixel.

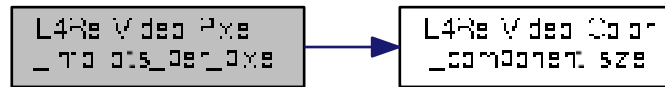
##### Returns

Number of bits used by the pixel.

Definition at line 147 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:



#### 14.285.3.6 bytes\_per\_pixel() [1/2]

```
unsigned char L4Re::Video::Pixel_info::bytes_per_pixel () const [inline]
```

Query size of pixel in bytes.

##### Returns

Size of pixel in bytes.

Definition at line [141](#) of file [colors](#).

#### 14.285.3.7 bytes\_per\_pixel() [2/2]

```
void L4Re::Video::Pixel_info::bytes_per_pixel (
 unsigned char bpp) [inline]
```

Set the size of the pixel in bytes.

##### Parameters

|            |                         |
|------------|-------------------------|
| <i>bpp</i> | Size of pixel in bytes. |
|------------|-------------------------|

Definition at line [184](#) of file [colors](#).

#### 14.285.3.8 dump()

```
template<typename OUT >
void L4Re::Video::Pixel_info::dump (
 OUT & s) const [inline]
```

Dump information on the pixel to a stream.

## Parameters

|   |        |
|---|--------|
| s | Stream |
|---|--------|

## Returns

The stream

Definition at line 238 of file [colors](#).

Referenced by [L4Re::Video::View::Info::dump\(\)](#).

Here is the caller graph for this function:

14.285.3.9 `g()` [1/2]

```
Color_component const& L4Re::Video::Pixel_info::g () const [inline]
```

Return the green color component of the pixel.

## Returns

Green color component.

Definition at line 123 of file [colors](#).

14.285.3.10 `g()` [2/2]

```
void L4Re::Video::Pixel_info::g (
 Color_component const & c) [inline]
```

Set the green color component of the pixel.

## Parameters

|                |                        |
|----------------|------------------------|
| <code>c</code> | Green color component. |
|----------------|------------------------|

Definition at line 166 of file [colors](#).

14.285.3.11 `has_alpha()`

```
bool L4Re::Video::Pixel_info::has_alpha () const [inline]
```

Return whether the pixel has an alpha channel.

## Returns

True if the pixel has an alpha channel, false if not.

Definition at line 154 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:

14.285.3.12 `operator==(`

```
bool L4Re::Video::Pixel_info::operator== (
 Pixel_info const & o) const [inline]
```

Compare for complete equality of the color space.

## Parameters

|                |                                             |
|----------------|---------------------------------------------|
| <code>o</code> | A <a href="#">Pixel_info</a> to compare to. |
|----------------|---------------------------------------------|

**Returns**

true if the both [Pixel\\_info](#)'s are equal, false if not.

Definition at line [227](#) of file [colors](#).

**14.285.3.13** [r\(\)](#) [1/2]

```
Color_component const& L4Re::Video::Pixel_info::r () const [inline]
```

Return the red color component of the pixel.

**Returns**

Red color component.

Definition at line [117](#) of file [colors](#).

**14.285.3.14** [r\(\)](#) [2/2]

```
void L4Re::Video::Pixel_info::r (
 Color_component const & c) [inline]
```

Set the red color component of the pixel.

**Parameters**

|          |                      |
|----------|----------------------|
| <b>c</b> | Red color component. |
|----------|----------------------|

Definition at line [160](#) of file [colors](#).

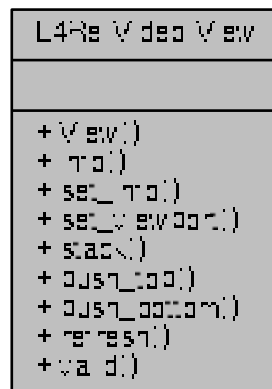
The documentation for this class was generated from the following file:

- [l4/re/video/colors](#)

## 14.286 L4Re::Video::View Class Reference

[View](#).

Collaboration diagram for L4Re::Video::View:



## Data Structures

- struct [Info](#)  
*Information structure of a view.*

## Public Types

- enum [Flags](#) {  
[F\\_none](#) = 0x00, [F\\_set\\_buffer](#) = 0x01, [F\\_set\\_buffer\\_offset](#) = 0x02, [F\\_set\\_bytes\\_per\\_line](#) = 0x04,  
[F\\_set\\_pixel](#) = 0x08, [F\\_set\\_position](#) = 0x10, [F\\_dyn\\_allocated](#) = 0x20, [F\\_set\\_background](#) = 0x40,  
[F\\_set\\_flags](#) = 0x80, [F\\_fully\\_dynamic](#) }  
*Flags on a view.*
- enum [V\\_flags](#) { [F\\_above](#) = 0x1000, [F\\_flags\\_mask](#) = 0xff000 }

*Property flags of a view.*

## Public Member Functions

- int [info](#) ([Info](#) \*info) const throw ()  
*Return the view information of the view.*
- int [set\\_info](#) ([Info](#) const &info) const throw ()  
*Set the information structure for this view.*
- int [set\\_viewport](#) (int scr\_x, int scr\_y, int w, int h, unsigned long buf\_offset) const throw ()  
*Set the position of the view in the goos.*
- int [stack](#) ([View](#) const &pivot, bool behind=true) const throw ()  
*Move this view in the view stack.*
- int [push\\_top](#) () const throw ()  
*Make this view the top-most view.*
- int [push\\_bottom](#) () const throw ()  
*Push this view the back.*
- int [refresh](#) (int x, int y, int w, int h) const throw ()  
*Refresh/Redraw the view.*
- bool [valid](#) () const  
*Return whether this view is valid.*

### 14.286.1 Detailed Description

[View](#).

Definition at line 34 of file [goos](#).

### 14.286.2 Member Enumeration Documentation

#### 14.286.2.1 Flags

enum [L4Re::Video::View::Flags](#)

Flags on a view.

Enumerator

|                                      |                                                                    |
|--------------------------------------|--------------------------------------------------------------------|
| <a href="#">F_none</a>               | everything for this view is static (the VESA-FB case)              |
| <a href="#">F_set_buffer</a>         | buffer object for this view can be changed                         |
| <a href="#">F_set_buffer_offset</a>  | buffer offset can be set                                           |
| <a href="#">F_set_bytes_per_line</a> | bytes per line can be set                                          |
| <a href="#">F_set_pixel</a>          | pixel type can be set                                              |
| <a href="#">F_set_position</a>       | position on screen can be set                                      |
| <a href="#">F_dyn_allocated</a>      | <a href="#">View</a> is dynamically allocated.                     |
| <a href="#">F_set_background</a>     | Set view as background for session.                                |
| <a href="#">F_set_flags</a>          | Set view flags (.<br><br>See also<br><br><a href="#">V_flags</a> ) |
| <a href="#">F_fully_dynamic</a>      | Flags for a fully dynamic view.                                    |

Definition at line 54 of file [goos](#).

#### 14.286.2.2 V\_flags

enum [L4Re::Video::View::V\\_flags](#)

Property flags of a view.

Such flags can be set or deleted with the [F\\_set\\_flags](#) operation using the [set\\_info\(\)](#) method.

Enumerator

|                              |                                              |
|------------------------------|----------------------------------------------|
| <a href="#">F_above</a>      | Flag the view as stay on top.                |
| <a href="#">F_flags_mask</a> | Mask containing all possible property flags. |

Definition at line 77 of file [goos](#).

## 14.286.3 Member Function Documentation

### 14.286.3.1 info()

```
int L4Re::Video::View::info (
 Info * info) const throw) [inline]
```

Return the view information of the view.

#### Parameters

|     |             |                                |
|-----|-------------|--------------------------------|
| out | <i>info</i> | Information structure pointer. |
|-----|-------------|--------------------------------|

#### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 351 of file [goos](#).

### 14.286.3.2 refresh()

```
int L4Re::Video::View::refresh (
 int x,
 int y,
 int w,
 int h) const throw) [inline]
```

Refresh/Redraw the view.

#### Parameters

|          |             |
|----------|-------------|
| <i>x</i> | X position. |
| <i>y</i> | Y position. |
| <i>w</i> | Width.      |
| <i>h</i> | Height.     |

#### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |



Definition at line 363 of file [goos](#).

### 14.286.3.3 set\_info()

```
int L4Re::Video::View::set_info (
 Info const & info) const throw () [inline]
```

Set the information structure for this view.

#### Parameters

|             |                        |
|-------------|------------------------|
| <i>info</i> | Information structure. |
|-------------|------------------------|

#### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

The function will also set the view port according to the values given in the information structure.

Definition at line 355 of file [goos](#).

### 14.286.3.4 set\_viewport()

```
int L4Re::Video::View::set_viewport (
 int scr_x,
 int scr_y,
 int w,
 int h,
 unsigned long buf_offset) const throw () [inline]
```

Set the position of the view in the goos.

#### Parameters

|                   |                               |
|-------------------|-------------------------------|
| <i>scr_x</i>      | X position                    |
| <i>scr_y</i>      | Y position                    |
| <i>w</i>          | Width                         |
| <i>h</i>          | Height                        |
| <i>buf_offset</i> | Offset in the buffer in bytes |

#### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 367 of file [goos](#).

References [L4Re::Video::View::Info::buffer\\_offset](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

#### 14.286.3.5 stack()

```
int L4Re::Video::View::stack (
 View const & pivot,
 bool behind = true) const throw () [inline]
```

Move this view in the view stack.

##### Parameters

|               |                                                                                              |
|---------------|----------------------------------------------------------------------------------------------|
| <i>pivot</i>  | <a href="#">View</a> to move relative to                                                     |
| <i>behind</i> | When true move the view behind the pivot view, if false move the view before the pivot view. |

##### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 359 of file [goos](#).

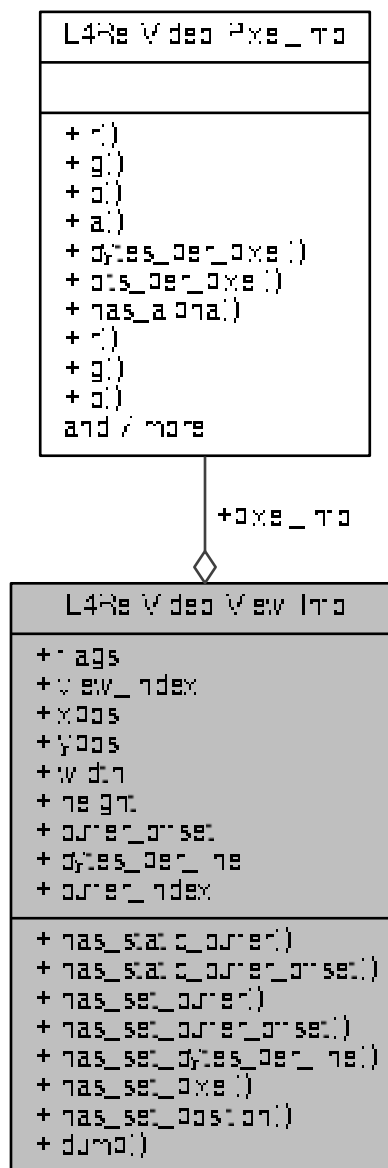
The documentation for this class was generated from the following file:

- [l4/re/video/goos](#)

## 14.287 L4Re::Video::View::Info Struct Reference

Information structure of a view.

Collaboration diagram for L4Re::Video::View::Info:



## Public Member Functions

- bool [has\\_static\\_buffer](#) () const  
*Return whether the view has a static buffer.*
- bool [has\\_static\\_buffer\\_offset](#) () const  
*Return whether the static buffer offset is available.*
- bool [has\\_set\\_buffer](#) () const  
*Return whether a buffer is set.*
- bool [has\\_set\\_buffer\\_offset](#) () const

- Return whether the given buffer offset is valid.*

    - bool [has\\_set\\_bytes\\_per\\_line](#) () const

*Return whether the given bytes-per-line value is valid.*

  - bool [has\\_set\\_pixel](#) () const
- Return whether the given pixel information is valid.*
- bool [has\\_set\\_position](#) () const
- Return whether the position information given is valid.*
- template<typename OUT >  
void [dump](#) (OUT &s) const
- Dump information on the view information to a stream.*

## Data Fields

- unsigned [flags](#)  
*Flags, see [Flags](#) and [V\\_flags](#).*
- unsigned [view\\_index](#)  
*Index of the view.*
- unsigned long [xpos](#)  
*X position in pixels of the view in the goos.*
- unsigned long [ypos](#)  
*Y position in pixels of the view in the goos.*
- unsigned long [width](#)  
*Width of the view in pixels.*
- unsigned long [height](#)  
*Height of the view in pixels.*
- unsigned long [buffer\\_offset](#)  
*Offset in the memory buffer in bytes.*
- unsigned long [bytes\\_per\\_line](#)  
*Bytes per line.*
- [Pixel\\_info](#) [pixel\\_info](#)  
*Pixel information.*
- unsigned [buffer\\_index](#)  
*Number of the buffer used for this view.*

### 14.287.1 Detailed Description

Information structure of a view.

Definition at line 86 of file [goos](#).

The documentation for this struct was generated from the following file:

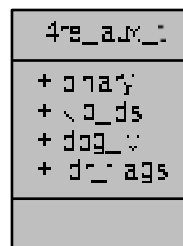
- [l4/re/video/goos](#)

## 14.288 l4re\_aux\_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

Collaboration diagram for l4re\_aux\_t:



### Data Fields

- `char const * binary`  
*Binary name.*
- `l4_cap_idx_t kip_ds`  
*Data space of the KIP.*
- `l4_umword_t dbg_lvl`  
*Debug levels for l4re.*
- `l4_umword_t ldr_flags`  
*Flags for l4re, see l4re\_aux\_ldr\_flags\_t.*

### 14.288.1 Detailed Description

Auxiliary descriptor.

Definition at line 51 of file [l4aux.h](#).

The documentation for this struct was generated from the following file:

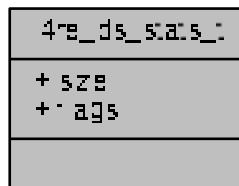
- [l4/re/l4aux.h](#)

## 14.289 l4re\_ds\_stats\_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re\_ds\_stats\_t:



### Data Fields

- unsigned long [size](#)  
*size*
- unsigned long [flags](#)  
*flags*

### 14.289.1 Detailed Description

Information about the data space.

Definition at line 45 of file [dataspace.h](#).

The documentation for this struct was generated from the following file:

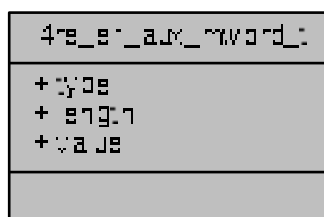
- [l4re/c/dataspace.h](#)

## 14.290 l4re\_elf\_aux\_mword\_t Struct Reference

Auxiliary vector element for a single unsigned data word.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_mword\_t:



### 14.290.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line 124 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

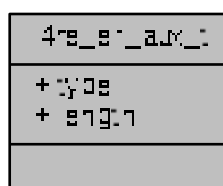
- [l4re/elf\\_aux.h](#)

## 14.291 l4re\_elf\_aux\_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_t:



### 14.291.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line 104 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

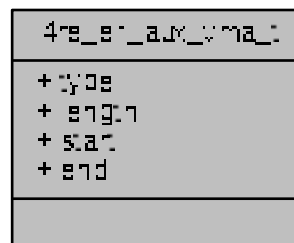
- [l4/re/elf\\_aux.h](#)

## 14.292 l4re\_elf\_aux\_vma\_t Struct Reference

Auxiliary vector element for a reserved virtual memory area.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_vma\_t:



### 14.292.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 113 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/elf\\_aux.h](#)

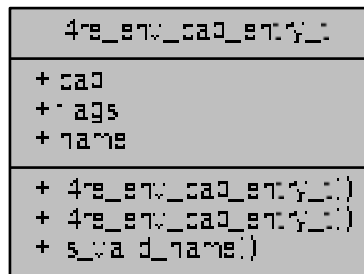


## 14.293 l4re\_env\_cap\_entry\_t Struct Reference

Entry in the [L4Re](#) environment array for the named initial objects.

```
#include <env.h>
```

Collaboration diagram for l4re\_env\_cap\_entry\_t:



### Public Member Functions

- [l4re\\_env\\_cap\\_entry\\_t\(\)](#)  
*Create an invalid entry.*
- [l4re\\_env\\_cap\\_entry\\_t](#) (char const \*n, [l4\\_cap\\_idx\\_t](#) c, [l4\\_umword\\_t](#) f=0)  
*Create an entry with the name n, capability c, and flags f.*

### Data Fields

- [l4\\_cap\\_idx\\_t](#) cap  
*The capability selector for the object.*
- [l4\\_umword\\_t](#) flags  
*Some flags for the object.*
- char [name](#) [16]  
*The name of the object.*

### 14.293.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

Definition at line [49](#) of file [env.h](#).

### 14.293.2 Constructor & Destructor Documentation

### 14.293.2.1 l4re\_env\_cap\_entry\_t()

```
l4re_env_cap_entry_t::l4re_env_cap_entry_t (
 char const * n,
 l4_cap_idx_t c,
 l4_umword_t f = 0) [inline]
```

Create an entry with the name *n*, capability *c*, and flags *f*.

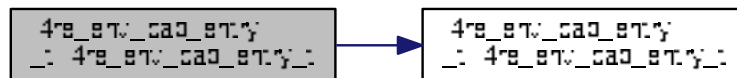
#### Parameters

|          |                                                            |
|----------|------------------------------------------------------------|
| <i>n</i> | is the name of the initial object.                         |
| <i>c</i> | is the capability selector that refers the initial object. |
| <i>f</i> | are the additional flags for the object.                   |

Definition at line 80 of file [env.h](#).

References [l4re\\_env\\_cap\\_entry\\_t\(\)](#), and [name](#).

Here is the call graph for this function:



## 14.293.3 Field Documentation

### 14.293.3.1 flags

```
l4_umword_t l4re_env_cap_entry_t::flags
```

Some flags for the object.

#### Note

Currently unused.

Definition at line 60 of file [env.h](#).

Referenced by [l4re\\_env\\_get\\_cap\\_l\(\)](#).

The documentation for this struct was generated from the following file:

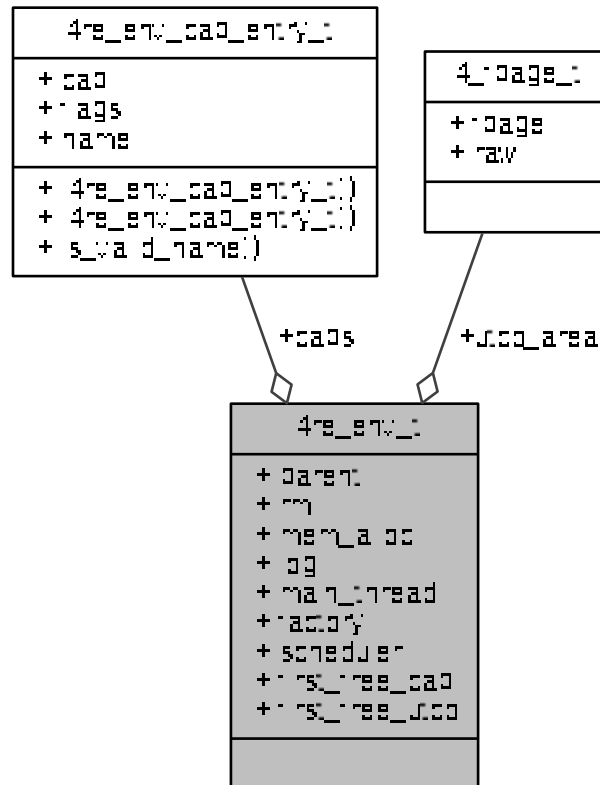
- [l4/re/env.h](#)

## 14.294 l4re\_env\_t Struct Reference

Initial environment data structure.

```
#include <env.h>
```

Collaboration diagram for l4re\_env\_t:



### Data Fields

- [l4\\_cap\\_idx\\_t parent](#)  
*Parent object-capability.*
- [l4\\_cap\\_idx\\_t rm](#)  
*Region map object-capability.*
- [l4\\_cap\\_idx\\_t mem\\_alloc](#)  
*Memory allocator object-capability.*
- [l4\\_cap\\_idx\\_t log](#)  
*Logging object-capability.*
- [l4\\_cap\\_idx\\_t main\\_thread](#)  
*Object-capability of the first user thread.*
- [l4\\_cap\\_idx\\_t factory](#)

*Object-capability of the factory available to the task.*

- [l4\\_cap\\_idx\\_t scheduler](#)

*Object capability for the scheduler set to use.*

- [l4\\_cap\\_idx\\_t first\\_free\\_cap](#)

*First capability index available to the application.*

- [l4\\_fpage\\_t utcb\\_area](#)

*UTCB area of the task.*

- [l4\\_addr\\_t first\\_free\\_utcb](#)

*First UTCB within the UTCB area available to the application.*

### 14.294.1 Detailed Description

Initial environment data structure.

See also

[Initial environment](#)

Definition at line 108 of file [env.h](#).

The documentation for this struct was generated from the following file:

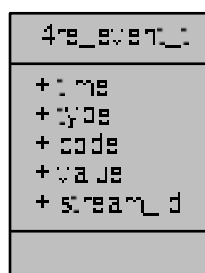
- [l4/re/env.h](#)

## 14.295 l4re\_event\_t Struct Reference

Event structure used in buffer.

```
#include <event.h>
```

Collaboration diagram for `l4re_event_t`:



## Data Fields

- long long [time](#)  
*Time stamp of the event.*
- unsigned short [type](#)  
*Type of the event.*
- unsigned short [code](#)  
*Code of the event.*
- int [value](#)  
*Value of the event.*
- [l4\\_umword\\_t](#) [stream\\_id](#)  
*Stream ID.*

### 14.295.1 Detailed Description

Event structure used in buffer.

Definition at line 40 of file [event.h](#).

The documentation for this struct was generated from the following file:

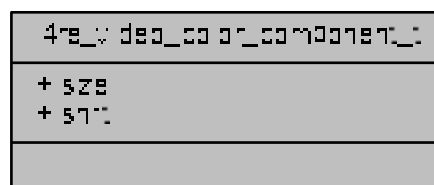
- [l4/re/c/event.h](#)

## 14.296 l4re\_video\_color\_component\_t Struct Reference

Color component structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_color\_component\_t:



## Data Fields

- unsigned char [size](#)  
*Size in bits.*
- unsigned char [shift](#)  
*offset in pixel*

### 14.296.1 Detailed Description

Color component structure.

Definition at line 31 of file [colors.h](#).

The documentation for this struct was generated from the following file:

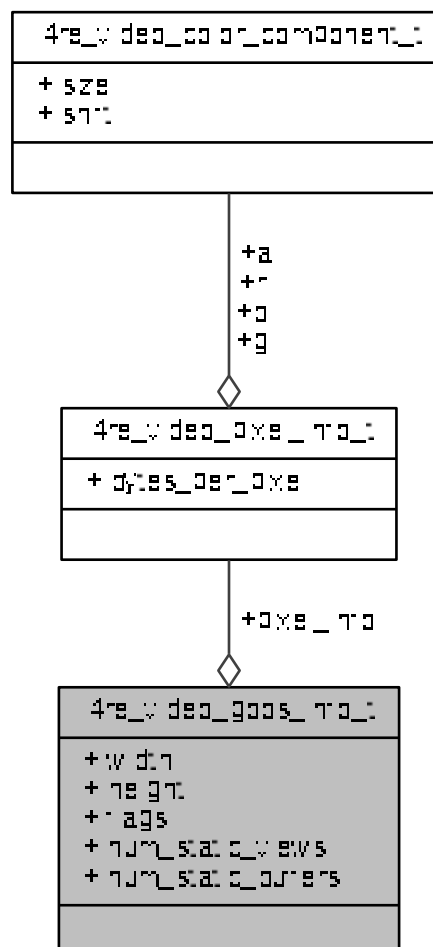
- [l4re/c/video/colors.h](#)

### 14.297 l4re\_video\_goos\_info\_t Struct Reference

Goos information structure.

```
#include <goos.h>
```

Collaboration diagram for l4re\_video\_goos\_info\_t:



## Data Fields

- unsigned long [width](#)  
*Width of the goos.*
- unsigned long [height](#)  
*Height of the goos.*
- unsigned [flags](#)  
*Flags of the framebuffer, see [l4re\\_video\\_goos\\_info\\_flags\\_t](#).*
- unsigned [num\\_static\\_views](#)  
*Number of static views.*
- unsigned [num\\_static\\_buffers](#)  
*Number of static buffers.*
- [l4re\\_video\\_pixel\\_info\\_t pixel\\_info](#)  
*Pixel layout of the goos.*

## 14.297.1 Detailed Description

Goos information structure.

Definition at line 51 of file [goos.h](#).

The documentation for this struct was generated from the following file:

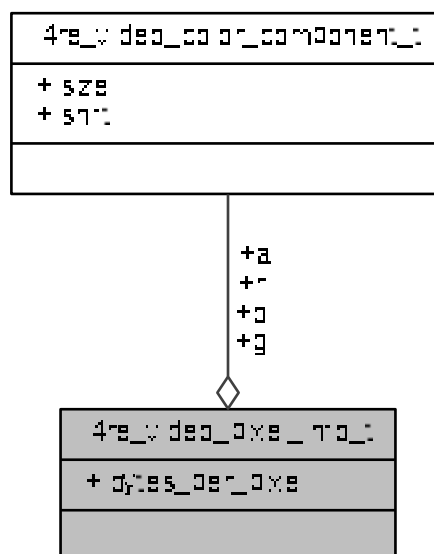
- [l4re/c/video/goos.h](#)

## 14.298 l4re\_video\_pixel\_info\_t Struct Reference

Pixel\_info structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_pixel\_info\_t:



## Data Fields

- [l4re\\_video\\_color\\_component\\_t](#) a

*Colors.*

- unsigned char [bytes\\_per\\_pixel](#)

*Bytes per pixel.*

### 14.298.1 Detailed Description

Pixel\_info structure.

Definition at line [41](#) of file [colors.h](#).

The documentation for this struct was generated from the following file:

- [l4re/c/video/colors.h](#)

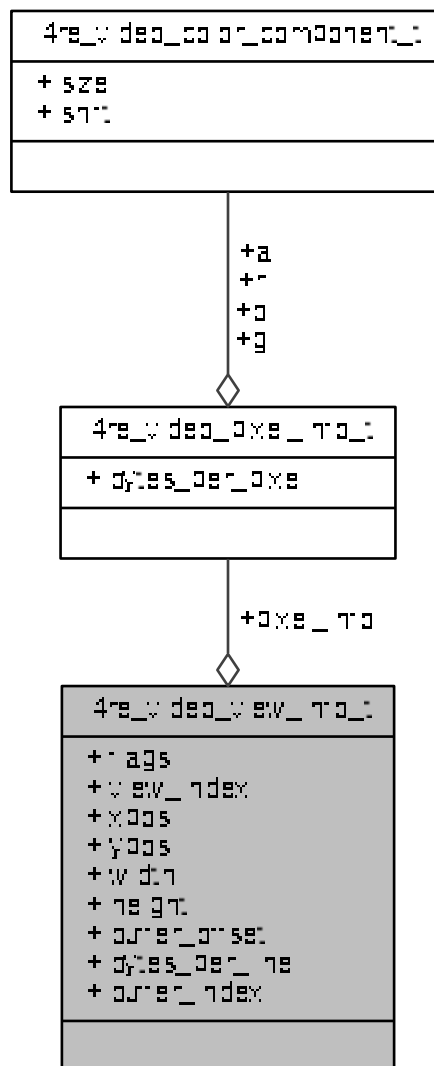
## 14.299 l4re\_video\_view\_info\_t Struct Reference

View information structure.

```
#include <view.h>
```



Collaboration diagram for l4re\_video\_view\_info\_t:



## Data Fields

- unsigned [flags](#)  
*Flags.*
- unsigned [view\\_index](#)  
*Number of view in the goos.*
- unsigned long [height](#)  
*Position in goos and size of view.*
- unsigned long [buffer\\_offset](#)  
*Memory offset in goos buffer.*
- unsigned long [bytes\\_per\\_line](#)  
*Size of line in view.*

- [l4re\\_video\\_pixel\\_info\\_t pixel\\_info](#)  
*Pixel info.*
- unsigned [buffer\\_index](#)  
*Number of buffer of goos.*

### 14.299.1 Detailed Description

View information structure.

Definition at line 59 of file [view.h](#).

The documentation for this struct was generated from the following file:

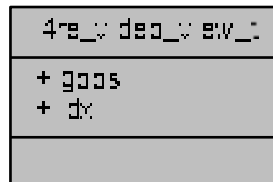
- [l4re/c/video/view.h](#)

## 14.300 l4re\_video\_view\_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for `l4re_video_view_t`:



### 14.300.1 Detailed Description

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

Definition at line 78 of file [view.h](#).

The documentation for this struct was generated from the following file:

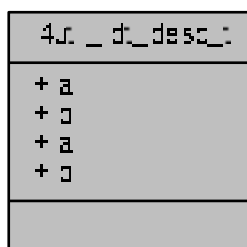
- [l4re/c/video/view.h](#)

## 14.301 l4util\_idt\_desc\_t Struct Reference

IDT entry.

```
#include <idt.h>
```

Collaboration diagram for l4util\_idt\_desc\_t:



### Data Fields

- [l4\\_uint64\\_t b](#)  
*see Intel doc*
- [l4\\_uint32\\_t b](#)  
*see Intel doc*

### 14.301.1 Detailed Description

IDT entry.

Definition at line 33 of file [idt.h](#).

The documentation for this struct was generated from the following file:

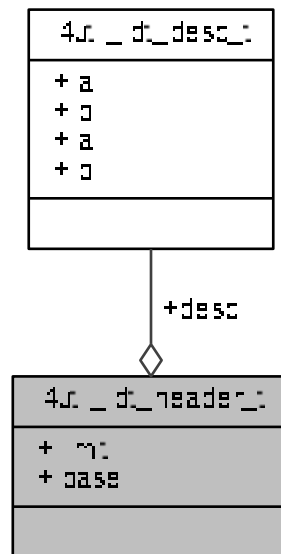
- `amd64/l4/util/idt.h`

## 14.302 l4util\_idt\_header\_t Struct Reference

Header of an IDT table.

```
#include <idt.h>
```

Collaboration diagram for l4util\_idt\_header\_t:



### Data Fields

- [l4\\_uint16\\_t limit](#)  
*limit field (see Intel doc)*
- void \* [base](#)  
*idt base (see Intel doc)*

### 14.302.1 Detailed Description

Header of an IDT table.

Definition at line 40 of file [idt.h](#).

The documentation for this struct was generated from the following file:

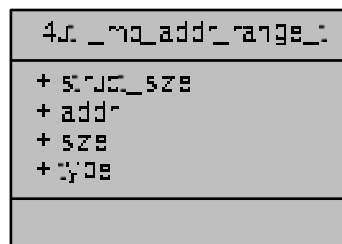
- amd64/l4/util/[idt.h](#)

## 14.303 l4util\_mb\_addr\_range\_t Struct Reference

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_addr\_range\_t:



### Data Fields

- [l4\\_uint32\\_t struct\\_size](#)  
*Size of structure.*
- [l4\\_uint64\\_t addr](#)  
*Start address.*
- [l4\\_uint64\\_t size](#)  
*Size of memory range.*
- [l4\\_uint32\\_t type](#)  
*type of memory range*

### 14.303.1 Detailed Description

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

Definition at line 47 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

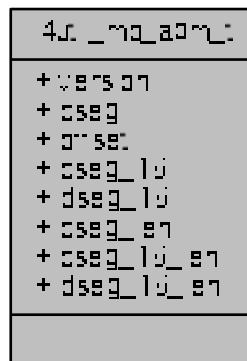
- [l4/util/mb\\_info.h](#)

## 14.304 l4util\_mb\_apm\_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_apm\_t:



### 14.304.1 Detailed Description

APM BIOS info.

Definition at line 95 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

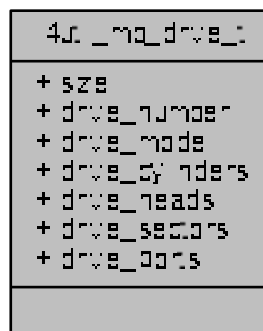
- [l4/util/mb\\_info.h](#)

## 14.305 l4util\_mb\_drive\_t Struct Reference

Drive Info structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_drive\_t:



## Data Fields

- [l4\\_uint8\\_t drive\\_number](#)  
< The size of this structure.
- [l4\\_uint8\\_t drive\\_mode](#)  
< The BIOS drive number.
- [l4\\_uint16\\_t drive\\_cylinders](#)  
< The access mode (see below).
- [l4\\_uint8\\_t drive\\_heads](#)  
< number of cylinders
- [l4\\_uint8\\_t drive\\_sectors](#)  
< number of heads
- [l4\\_uint16\\_t drive\\_ports](#) [0]  
< number of sectors per track

## 14.305.1 Detailed Description

Drive Info structure.

Definition at line 78 of file [mb\\_info.h](#).

## 14.305.2 Field Documentation

### 14.305.2.1 drive\_cylinders

[l4\\_uint16\\_t](#) l4util\_mb\_drive\_t::drive\_cylinders

<The access mode (see below).

Definition at line 83 of file [mb\\_info.h](#).

## 14.305.2.2 drive\_mode

```
l4_uint8_t l4util_mb_drive_t::drive_mode
```

<The BIOS drive number.

Definition at line 82 of file [mb\\_info.h](#).

## 14.305.2.3 drive\_number

```
l4_uint8_t l4util_mb_drive_t::drive_number
```

<The size of this structure.

Definition at line 81 of file [mb\\_info.h](#).

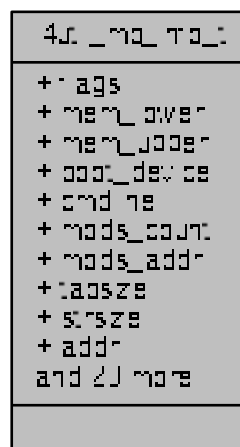
The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 14.306 l4util\_mb\_info\_t Struct Reference

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_info\_t:





## Data Fields

- [l4\\_uint32\\_t flags](#)  
*MultiBoot info version number.*
- [l4\\_uint32\\_t mem\\_lower](#)  
*available memory below 1MB*
- [l4\\_uint32\\_t mem\\_upper](#)  
*available memory starting from 1MB [kB]*
- [l4\\_uint32\\_t boot\\_device](#)  
*"root" partition*
- [l4\\_uint32\\_t cmdline](#)  
*Kernel command line.*
- [l4\\_uint32\\_t mods\\_count](#)  
*number of modules*
- [l4\\_uint32\\_t mods\\_addr](#)  
*module list*
- [l4\\_uint32\\_t mmap\\_length](#)  
*size of memory mapping buffer*
- [l4\\_uint32\\_t mmap\\_addr](#)  
*address of memory mapping buffer*
- [l4\\_uint32\\_t drives\\_length](#)  
*size of drive info buffer*
- [l4\\_uint32\\_t drives\\_addr](#)  
*address of driver info buffer*
- [l4\\_uint32\\_t config\\_table](#)  
*ROM configuration table.*
- [l4\\_uint32\\_t boot\\_loader\\_name](#)  
*Boot Loader Name.*
- [l4\\_uint32\\_t apm\\_table](#)  
*APM table.*
- [l4\\_uint32\\_t vbe\\_ctrl\\_info](#)  
*VESA video controller info.*
- [l4\\_uint32\\_t vbe\\_mode\\_info](#)  
*VESA video mode info.*
- [l4\\_uint16\\_t vbe\\_mode](#)  
*VESA video mode number.*
- [l4\\_uint16\\_t vbe\\_interface\\_seg](#)  
*VESA segment of prot BIOS interface.*
- [l4\\_uint16\\_t vbe\\_interface\\_off](#)  
*VESA offset of prot BIOS interface.*
- [l4\\_uint16\\_t vbe\\_interface\\_len](#)  
*VESA length of prot BIOS interface.*
- [l4\\_uint32\\_t tabsize](#)  
*(a.out) Kernel symbol table info*
- [l4\\_uint32\\_t num](#)  
*(ELF) Kernel section header table*

### 14.306.1 Detailed Description

MultiBoot Info description

This is the struct passed to the boot image. This is done by placing its address in the EAX register.

Definition at line 207 of file [mb\\_info.h](#).

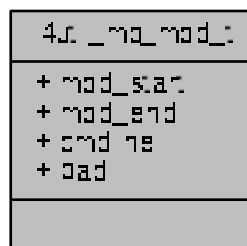
The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

### 14.307 l4util\_mb\_mod\_t Struct Reference

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_mod\_t:



#### Data Fields

- [l4\\_uint32\\_t mod\\_start](#)  
*Starting address of module in memory.*
- [l4\\_uint32\\_t mod\\_end](#)  
*End address of module in memory.*
- [l4\\_uint32\\_t cmdline](#)  
*Module command line.*
- [l4\\_uint32\\_t pad](#)  
*padding to take it to 16 bytes*

### 14.307.1 Detailed Description

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

Definition at line 31 of file [mb\\_info.h](#).

## 14.307.2 Field Documentation

### 14.307.2.1 mod\_end

`l4_uint32_t l4util_mb_mod_t::mod_end`

End address of module in memory.

Definition at line 34 of file [mb\\_info.h](#).

### 14.307.2.2 mod\_start

`l4_uint32_t l4util_mb_mod_t::mod_start`

Starting address of module in memory.

Definition at line 33 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

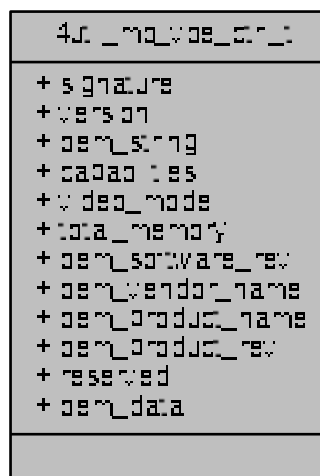
- [l4/util/mb\\_info.h](#)

## 14.308 l4util\_mb\_vbe\_ctrl\_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_vbe_ctrl_t`:



### 14.308.1 Detailed Description

VBE controller information.

Definition at line [109](#) of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 14.309 l4util\_mb\_vbe\_mode\_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

[illegible]

## all VESA versions

- `l4_uint16_t` `mode_attributes`
- `l4_uint8_t` `win_a_attributes`
- `l4_uint8_t` `win_b_attributes`
- `l4_uint16_t` `win_granularity`

- [l4\\_uint16\\_t](#) win\_size
- [l4\\_uint16\\_t](#) win\_a\_segment
- [l4\\_uint16\\_t](#) win\_b\_segment
- [l4\\_uint32\\_t](#) win\_func
- [l4\\_uint16\\_t](#) bytes\_per\_scanline

#### >= VESA version 1.2

- [l4\\_uint16\\_t](#) x\_resolution
- [l4\\_uint16\\_t](#) y\_resolution
- [l4\\_uint8\\_t](#) x\_char\_size
- [l4\\_uint8\\_t](#) y\_char\_size
- [l4\\_uint8\\_t](#) number\_of\_planes
- [l4\\_uint8\\_t](#) bits\_per\_pixel
- [l4\\_uint8\\_t](#) number\_of\_banks
- [l4\\_uint8\\_t](#) memory\_model
- [l4\\_uint8\\_t](#) bank\_size
- [l4\\_uint8\\_t](#) number\_of\_image\_pages
- [l4\\_uint8\\_t](#) reserved0

#### direct color

- [l4\\_uint8\\_t](#) red\_mask\_size
- [l4\\_uint8\\_t](#) red\_field\_position
- [l4\\_uint8\\_t](#) green\_mask\_size
- [l4\\_uint8\\_t](#) green\_field\_position
- [l4\\_uint8\\_t](#) blue\_mask\_size
- [l4\\_uint8\\_t](#) blue\_field\_position
- [l4\\_uint8\\_t](#) reserved\_mask\_size
- [l4\\_uint8\\_t](#) reserved\_field\_position
- [l4\\_uint8\\_t](#) direct\_color\_mode\_info

#### >= VESA version 2.0

- [l4\\_uint32\\_t](#) phys\_base
- [l4\\_uint32\\_t](#) reserved1
- [l4\\_uint16\\_t](#) reversed2

#### >= VESA version 3.0

- [l4\\_uint16\\_t](#) linear\_bytes\_per\_scanline
- [l4\\_uint8\\_t](#) banked\_number\_of\_image\_pages
- [l4\\_uint8\\_t](#) linear\_number\_of\_image\_pages
- [l4\\_uint8\\_t](#) linear\_red\_mask\_size
- [l4\\_uint8\\_t](#) linear\_red\_field\_position
- [l4\\_uint8\\_t](#) linear\_green\_mask\_size
- [l4\\_uint8\\_t](#) linear\_green\_field\_position
- [l4\\_uint8\\_t](#) linear\_blue\_mask\_size
- [l4\\_uint8\\_t](#) linear\_blue\_field\_position
- [l4\\_uint8\\_t](#) linear\_reserved\_mask\_size
- [l4\\_uint8\\_t](#) linear\_reserved\_field\_position
- [l4\\_uint32\\_t](#) max\_pixel\_clock
- [l4\\_uint8\\_t](#) reserved3 [189+1]

### 14.309.1 Detailed Description

VBE mode information.

Definition at line 127 of file [mb\\_info.h](#).

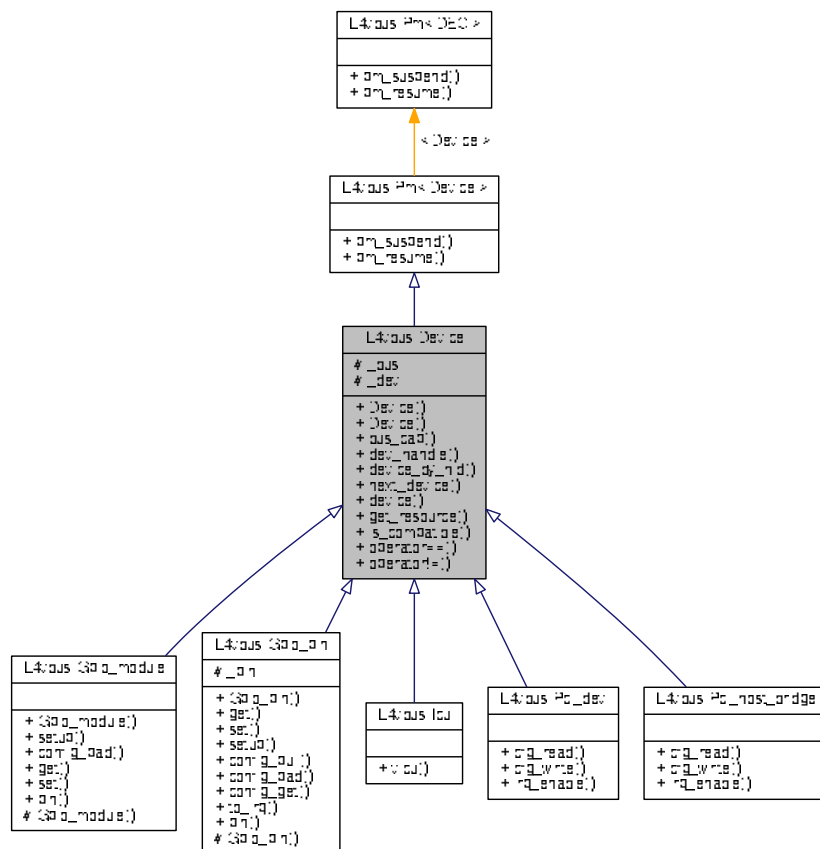
The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

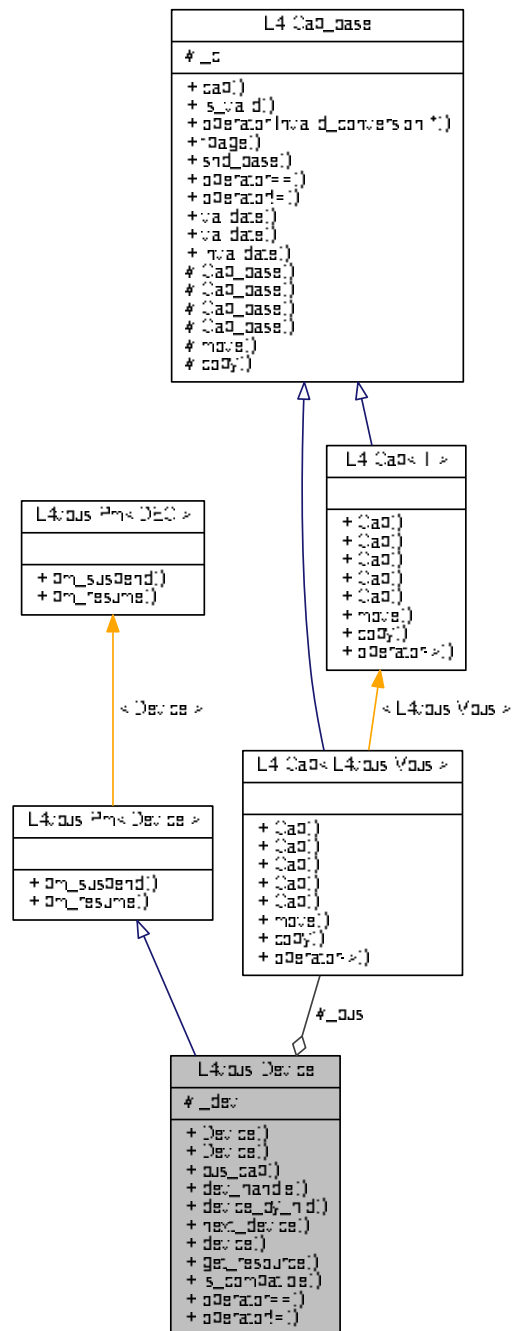
## 14.310 L4vbus::Device Class Reference

Device on a [L4vbus::Vbus](#).

Inheritance diagram for L4vbus::Device:



Collaboration diagram for L4vbus::Device:



## Public Member Functions

- **L4::Cap< Vbus > bus\_cap () const**  
Access the *Vbus* capability of the underlying virtual bus.
- **l4vbus\_device\_handle\_t dev\_handle () const**  
Access the device handle of this device.



- int [device\\_by\\_hid](#) ([Device](#) \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find a device by the human interface identifier (HID).*
- int [next\\_device](#) ([Device](#) \*child, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find next child following *child*.*
- int [device](#) ([l4vbus\\_device\\_t](#) \*devinfo) const  
*Obtain detailed information about a *Vbus* device.*
- int [get\\_resource](#) (int res\_idx, [l4vbus\\_resource\\_t](#) \*res) const  
*Obtain the resource description of an individual device resource.*
- int [is\\_compatible](#) (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches *cid*.*
- bool [operator==](#) ([Device](#) const &o) const  
*Test if two devices are the same *Vbus* device.*
- bool [operator!=](#) ([Device](#) const &o) const  
*Test if two devices are not the same.*

## Protected Attributes

- [L4::Cap< Vbus > \\_bus](#)
- [l4vbus\\_device\\_handle\\_t \\_dev](#)  
*The device handle for this device.*

### 14.310.1 Detailed Description

[Device](#) on a [L4vbus::Vbus](#).

Definition at line 71 of file [vbus](#).

### 14.310.2 Member Function Documentation

#### 14.310.2.1 [bus\\_cap\(\)](#)

```
L4::Cap<Vbus> L4vbus::Device::bus_cap () const [inline]
```

Access the [Vbus](#) capability of the underlying virtual bus.

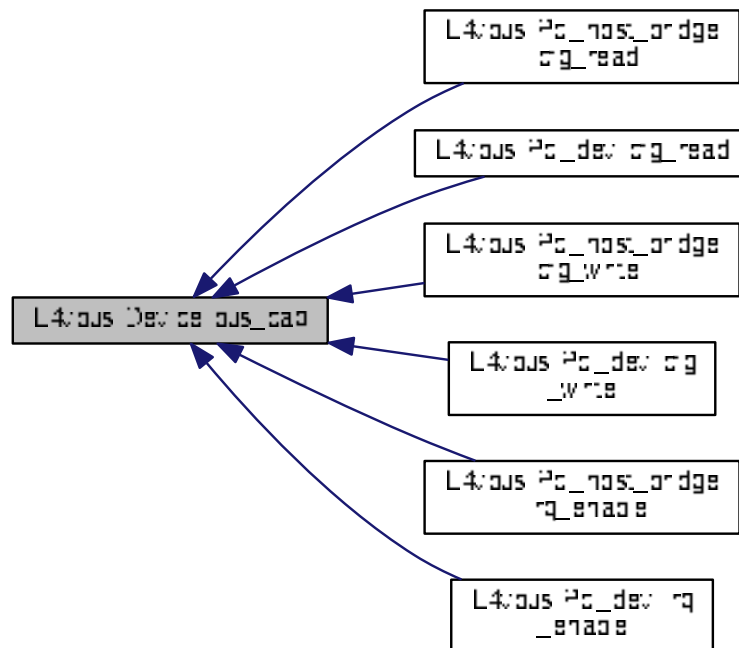
**Returns**

the capability to the underlying [Vbus](#).

Definition at line 83 of file [vbus](#).

Referenced by [L4vbus::Pci\\_host\\_bridge::cfg\\_read\(\)](#), [L4vbus::Pci\\_dev::cfg\\_read\(\)](#), [L4vbus::Pci\\_host\\_bridge::cfg\\_write\(\)](#), [L4vbus::Pci\\_dev::cfg\\_write\(\)](#), [L4vbus::Pci\\_host\\_bridge::irq\\_enable\(\)](#), and [L4vbus::Pci\\_dev::irq\\_enable\(\)](#).

Here is the caller graph for this function:

**14.310.2.2 dev\_handle()**

```
l4vbus_device_handle_t L4vbus::Device::dev_handle () const [inline]
```

Access the device handle of this device.

**Returns**

the device handle for this device.

The device handle is used to directly address the device on its virtual bus.

Definition at line 92 of file [vbus](#).

### 14.310.2.3 device()

```
int L4vbus::Device::device (
 l4vbus_device_t * devinfo) const [inline]
```

Obtain detailed information about a [Vbus](#) device.

## Parameters

|     |                |                                                                                                                   |
|-----|----------------|-------------------------------------------------------------------------------------------------------------------|
| out | <i>devinfo</i> | Information structure which contains details about the device. The pointer might be NULL after a successful call. |
|-----|----------------|-------------------------------------------------------------------------------------------------------------------|

## Return values

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| 0          | Success.                                                          |
| -L4_ENODEV | No device with the given device handle <i>dev</i> could be found. |

Definition at line 164 of file [vbus](#).

References [l4vbus\\_get\\_device\(\)](#).

Here is the call graph for this function:



## 14.310.2.4 device\_by\_hid()

```

int L4vbus::Device::device_by_hid (
 Device * child,
 char const * hid,
 int depth = L4VBUS_MAX_DEPTH,
 l4vbus_device_t * devinfo = 0) const [inline]

```

Find a device by the human interface identifier (HID).

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with *child* pointing to the device found in the previous iteration. The iteration starts at *child* that might be any device node in the tree.

## Parameters

|         |                |                                                                                                                                                                                                                                                                                    |
|---------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>child</i>   | Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). If a matching device is found its handle is returned through this parameter. |
|         | <i>hid</i>     | HID of the device                                                                                                                                                                                                                                                                  |
|         | <i>depth</i>   | Maximum depth for the recursive lookup                                                                                                                                                                                                                                             |
| out     | <i>devinfo</i> | <a href="#">Device</a> information structure (might be NULL)                                                                                                                                                                                                                       |

## Return values

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>&gt;=</code>      | 0 A device with the given HID was found.                 |
| <code>-L4_ENOENT</code> | No device with the given HID could be found on the vbus. |
| <code>-L4_EINVAL</code> | Invalid or no HID provided.                              |
| <code>-L4_ENODEV</code> | Function called on a non-existing device.                |

Definition at line 124 of file [vbus](#).

14.310.2.5 `get_resource()`

```
int L4vbus::Device::get_resource (
 int res_idx,
 l4vbus_resource_t * res) const [inline]
```

Obtain the resource description of an individual device resource.

## Parameters

|     |                |                                                                                                                                                                                                                                                                                                               |
|-----|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>res_idx</i> | Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the <a href="#">l4vbus_device_t</a> structure that is returned by <a href="#">L4vbus::Device::device_by_hid()</a> and <a href="#">L4vbus::Device::next_device()</a> . |
| out | <i>res</i>     | Descriptor of the resource.                                                                                                                                                                                                                                                                                   |

This function returns the resource descriptor of an individual device resource selected by the `res_idx` parameter.

## Return values

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>0</code>          | Success.                                      |
| <code>-L4_ENOENT</code> | Invalid resource index <code>res_idx</code> . |

Definition at line 184 of file [vbus](#).

References [l4vbus\\_get\\_resource\(\)](#).

Here is the call graph for this function:



### 14.310.2.6 is\_compatible()

```
int L4vbus::Device::is_compatible (
 char const * cid) const [inline]
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

#### Parameters

|            |                              |
|------------|------------------------------|
| <i>cid</i> | the compatibility ID to test |
|------------|------------------------------|

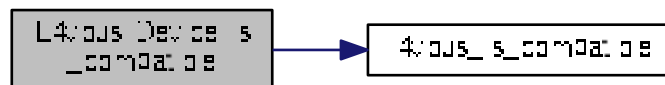
#### Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Definition at line 198 of file [vbus](#).

References [l4vbus\\_is\\_compatible\(\)](#).

Here is the call graph for this function:



### 14.310.2.7 next\_device()

```
int L4vbus::Device::next_device (
 Device * child,
 int depth = L4VBUS_MAX_DEPTH,
 l4vbus_device_t * devinfo = 0) const [inline]
```

Find next child following *child*.

#### Parameters

|         |                |                                                                                                                                                                                    |
|---------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>child</i>   | Handle of the device that precedes the device that shall be found. To start from the beginning <i>child</i> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). |
|         | <i>depth</i>   | Depth to look for                                                                                                                                                                  |
| out     | <i>devinfo</i> | device information (might be NULL)                                                                                                                                                 |

**Returns**

0 on success, else failure

Definition at line 145 of file [vbus](#).

**14.310.2.8 operator!=()**

```
bool L4vbus::Device::operator!= (
 Device const & o) const [inline]
```

Test if two devices are not the same.

**Returns**

true if the two devices are different, false else.

Definition at line 214 of file [vbus](#).

References [\\_bus](#), and [\\_dev](#).

**14.310.2.9 operator==()**

```
bool L4vbus::Device::operator== (
 Device const & o) const [inline]
```

Test if two devices are the same [Vbus](#) device.

**Returns**

true if the two devices are the same, false else.

Definition at line 205 of file [vbus](#).

References [\\_bus](#), and [\\_dev](#).

**14.310.3 Field Documentation**

### 14.310.3.1 `_bus`

`L4::Cap<Vbus> L4vbus::Device::_bus` [protected]

The `Vbus` capability (where this device is located on).

Definition at line 220 of file `vbus`.

Referenced by `L4vbus::Gpio_pin::config_get()`, `L4vbus::Gpio_pin::config_pad()`, `L4vbus::Gpio_module::config_pad()`, `L4vbus::Gpio_pin::config_pull()`, `L4vbus::Gpio_pin::get()`, `L4vbus::Gpio_module::get()`, `operator!=()`, `operator==()`, `L4vbus::Gpio_pin::set()`, `L4vbus::Gpio_module::set()`, `L4vbus::Gpio_pin::setup()`, `L4vbus::Gpio_module::setup()`, and `L4vbus::Gpio_pin::to_irq()`.

The documentation for this class was generated from the following file:

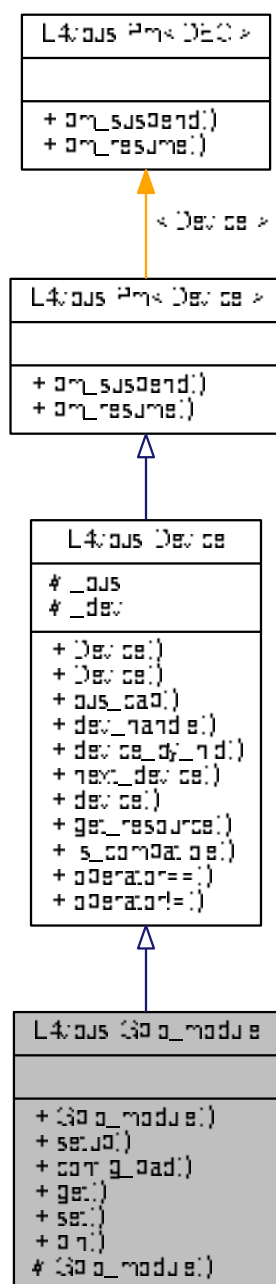
- `I4/vbus/vbus`

## 14.311 `L4vbus::Gpio_module` Class Reference

A `Gpio_module` groups multiple GPIO pins together.



Inheritance diagram for L4vbus::Gpio\_module:





- A slice of the pins provided by this module.*

## Public Member Functions

- `int setup (Pin_slice const &mask, unsigned mode, unsigned value) const`  
*Configure function of multiple GPIO pins at once.*
- `int config_pad (Pin_slice const &mask, unsigned func, unsigned value) const`  
*Hardware specific configuration function for multiple GPIO pins.*
- `int get (unsigned offset, unsigned *data) const`  
*Read values of multiple GPIO pins at once.*
- `int set (Pin_slice const &mask, unsigned data)`  
*Set multiple GPIO output pins at once.*
- `Gpio_pin pin (unsigned pin) const`  
*Get Gpio\_pin for a specific pin of this Gpio\_module.*

## Additional Inherited Members

### 14.311.1 Detailed Description

A [Gpio\\_module](#) groups multiple GPIO pins together.

Definition at line 129 of file [vbus\\_gpio](#).

### 14.311.2 Member Function Documentation

#### 14.311.2.1 config\_pad()

```
int L4vbus::Gpio_module::config_pad (
 Pin_slice const & mask,
 unsigned func,
 unsigned value) const [inline]
```

Hardware specific configuration function for multiple GPIO pins.

#### Parameters

|              |                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mask</i>  | Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation. |
| <i>func</i>  | Hardware specific configuration register, usually offset to the GPIO chip's base address.                                                                                                  |
| <i>value</i> | Value which is written into the hardware specific configuration register for the specified pins                                                                                            |

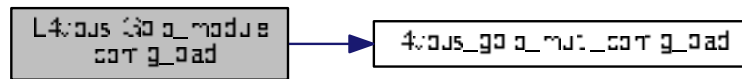
#### Returns

0 if OK, error code otherwise

Definition at line 181 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_config\\_pad\(\)](#).

Here is the call graph for this function:



#### 14.311.2.2 get()

```
int L4vbus::Gpio_module::get (
 unsigned offset,
 unsigned * data) const [inline]
```

Read values of multiple GPIO pins at once.

##### Parameters

|     |               |                                                                                                                             |
|-----|---------------|-----------------------------------------------------------------------------------------------------------------------------|
|     | <i>offset</i> | Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.                                       |
| out | <i>data</i>   | Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined. |

##### Returns

0 if OK, error code otherwise

Definition at line 197 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_get\(\)](#).

Here is the call graph for this function:



#### 14.311.2.3 pin()

```
Gpio_pin L4vbus::Gpio_module::pin (
 unsigned pin) const [inline]
```

Get [Gpio\\_pin](#) for a specific pin of this [Gpio\\_module](#).

## Parameters

|            |                                                      |
|------------|------------------------------------------------------|
| <i>pin</i> | GPIO pin number to get <a href="#">Gpio_pin</a> for. |
|------------|------------------------------------------------------|

## Returns

[Gpio\\_pin](#)

Definition at line 225 of file [vbus\\_gpio](#).

## 14.311.2.4 set()

```
int L4vbus::Gpio_module::set (
 Pin_slice const & mask,
 unsigned data) [inline]
```

Set multiple GPIO output pins at once.

## Parameters

|             |                                                                                                                                                                            |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mask</i> | Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation. |
| <i>data</i> | Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set.                                   |

## Returns

0 if OK, error code otherwise

Definition at line 213 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_set\(\)](#).

Here is the call graph for this function:



### 14.311.2.5 setup()

```
int L4vbus::Gpio_module::setup (
 Pin_slice const & mask,
 unsigned mode,
 unsigned value) const [inline]
```

Configure function of multiple GPIO pins at once.

#### Parameters

|              |                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mask</i>  | Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation. |
| <i>mode</i>  | GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.                                       |
| <i>value</i> | Optional value to set the GPIO pins to if they are configured as output pins                                                                                                               |

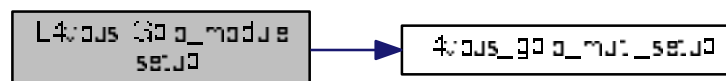
#### Returns

0 if OK, error code otherwise

Definition at line 162 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_setup\(\)](#).

Here is the call graph for this function:



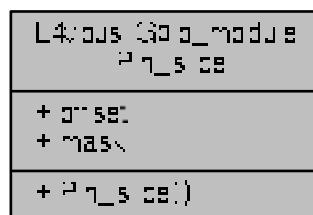
The documentation for this class was generated from the following file:

- `I4/vbus/vbus_gpio`

## 14.312 L4vbus::Gpio\_module::Pin\_slice Struct Reference

A slice of the pins provided by this module.

Collaboration diagram for L4vbus::Gpio\_module::Pin\_slice:



### 14.312.1 Detailed Description

A slice of the pins provided by this module.

Data type to specify a selection of pins for the 'multi' methods.

Definition at line 142 of file [vbus\\_gpio](#).

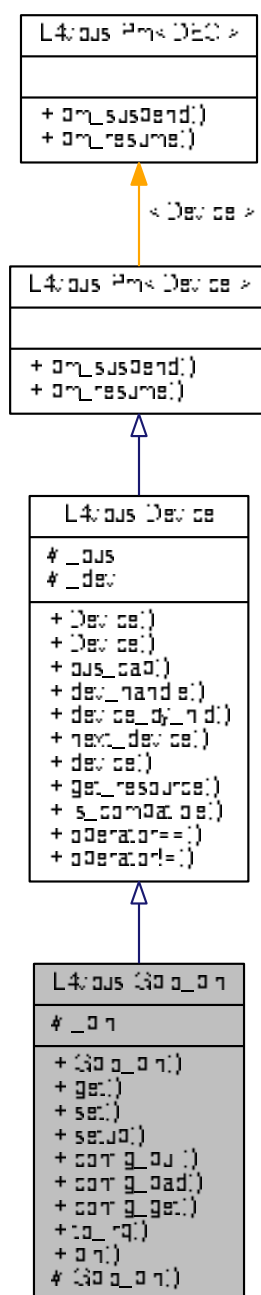
The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_gpio](#)

## 14.313 L4vbus::Gpio\_pin Class Reference

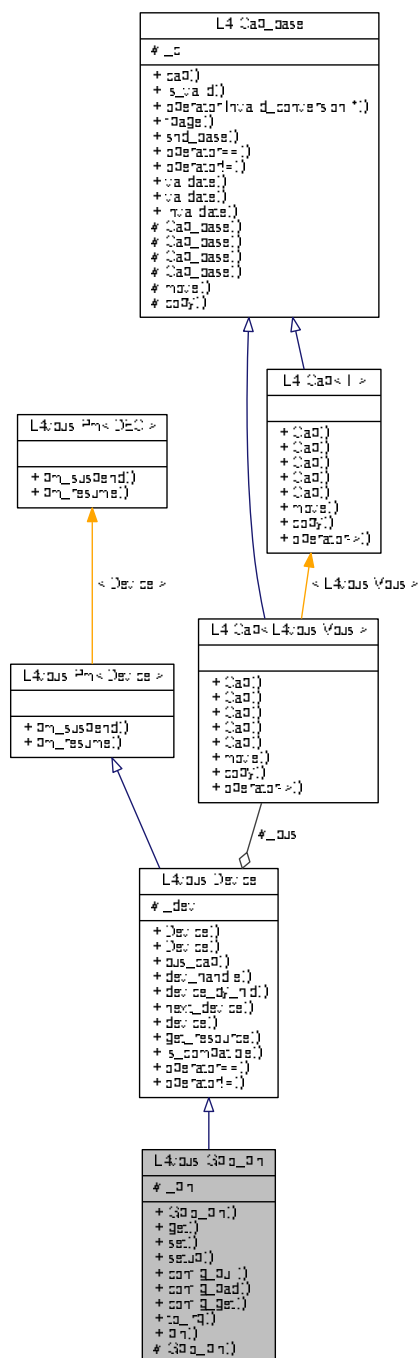
A GPIO pin.

Inheritance diagram for L4vbus::Gpio\_pin:





Collaboration diagram for L4vbus::Gpio\_pin:



## Public Member Functions

- int **get** () const  
*Read value of GPIO input pin.*
- int **set** (int value) const  
*Set GPIO output pin.*
- int **setup** (unsigned mode, unsigned value) const

*Configure the function of a GPIO pin.*

- int [config\\_pull](#) (unsigned mode) const

*Generic function to set pull up/down mode.*

- int [config\\_pad](#) (unsigned func, unsigned value) const

*Hardware specific configuration function.*

- int [config\\_get](#) (unsigned func, unsigned \*value) const

*Read hardware specific configuration.*

- int [to\\_irq](#) () const

*Create IRQ for GPIO pin.*

- unsigned [pin](#) () const

*Get pin number.*

## Additional Inherited Members

### 14.313.1 Detailed Description

A GPIO pin.

Definition at line 22 of file [vbus\\_gpio](#).

### 14.313.2 Member Function Documentation

#### 14.313.2.1 [config\\_get\(\)](#)

```
int L4vbus::Gpio_pin::config_get (
 unsigned func,
 unsigned * value) const [inline]
```

Read hardware specific configuration.

#### Parameters

|     |              |                                                                                                                   |
|-----|--------------|-------------------------------------------------------------------------------------------------------------------|
|     | <i>func</i>  | Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address. |
| out | <i>value</i> | The configuration value.                                                                                          |

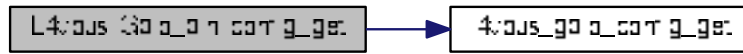
#### Returns

0 if OK, error code otherwise

Definition at line 98 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_get\(\)](#).

Here is the call graph for this function:



#### 14.313.2.2 config\_pad()

```
int L4vbus::Gpio_pin::config_pad (
 unsigned func,
 unsigned value) const [inline]
```

Hardware specific configuration function.

##### Parameters

|              |                                                                                                |
|--------------|------------------------------------------------------------------------------------------------|
| <i>func</i>  | Hardware specific configuration register, usually offset to the GPIO chip's base address       |
| <i>value</i> | Value which is written into the hardware specific configuration register for the specified pin |

##### Returns

0 if OK, error code otherwise

Definition at line 85 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pad\(\)](#).

Here is the call graph for this function:



#### 14.313.2.3 config\_pull()

```
int L4vbus::Gpio_pin::config_pull (
 unsigned mode) const [inline]
```

Generic function to set pull up/down mode.

**Parameters**

|             |                                                                             |
|-------------|-----------------------------------------------------------------------------|
| <i>mode</i> | mode for pull up/down resistors, see <a href="#">L4vbus_gpio_pull_modes</a> |
|-------------|-----------------------------------------------------------------------------|

**Returns**

0 if OK, error code otherwise

Definition at line 71 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pull\(\)](#).

Here is the call graph for this function:

**14.313.2.4 get()**

```
int L4vbus::Gpio_pin::get () const [inline]
```

Read value of GPIO input pin.

**Returns**

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Definition at line 34 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_get\(\)](#).

Here is the call graph for this function:



## 14.313.2.5 pin()

```
unsigned L4vbus::Gpio_pin::pin () const [inline]
```

Get pin number.

## Returns

GPIO pin number

Definition at line 118 of file [vbus\\_gpio](#).

## 14.313.2.6 set()

```
int L4vbus::Gpio_pin::set (
 int value) const [inline]
```

Set GPIO output pin.

## Parameters

|              |                                                 |
|--------------|-------------------------------------------------|
| <i>value</i> | Value to write to the GPIO pin (usually 0 or 1) |
|--------------|-------------------------------------------------|

## Returns

0 if OK, error code otherwise

Definition at line 45 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_set\(\)](#).

Here is the call graph for this function:



## 14.313.2.7 setup()

```
int L4vbus::Gpio_pin::setup (
 unsigned mode,
 unsigned value) const [inline]
```

Configure the function of a GPIO pin.

## Parameters

|              |                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mode</i>  | GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits. |
| <i>value</i> | Optional value to set the GPIO pin to if it is configured as an output pin                                                                           |

## Returns

0 if OK, error code otherwise

Definition at line 60 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_setup\(\)](#).

Here is the call graph for this function:



## 14.313.2.8 to\_irq()

```
int L4vbus::Gpio_pin::to_irq () const [inline]
```

Create IRQ for GPIO pin.

## Returns

IRQ number if OK, negative error code otherwise

Definition at line 108 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_to\\_irq\(\)](#).

Here is the call graph for this function:



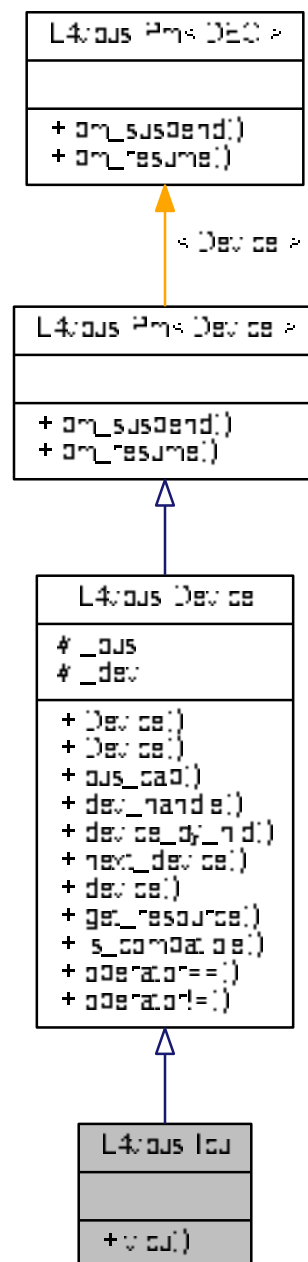
The documentation for this class was generated from the following file:

- [l4/vbus/vbus\\_gpio](#)

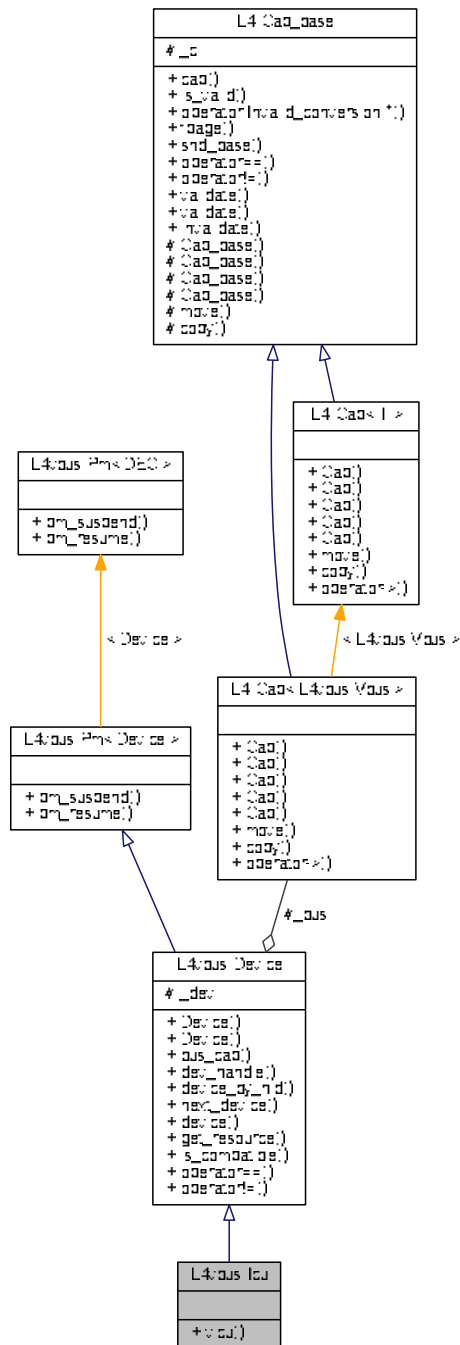
## 14.314 L4vbus::lcu Class Reference

[Vbus](#) Interrupt controller API.

Inheritance diagram for L4vbus::lcu:



Collaboration diagram for L4vbus::Icu:



## Public Member Functions

- `int vbus (L4::Cap< L4::Icu > icu) const`

*Request the L4::Icu capability for this Vbus Icu.*



## Additional Inherited Members

### 14.314.1 Detailed Description

[Vbus](#) Interrupt controller API.

Allows to access the underlying [L4::Icu](#) capability managing IRQs for the [L4vbus::Vbus](#).

See also

[L4::Icu](#)

Definition at line [234](#) of file [vbus](#).

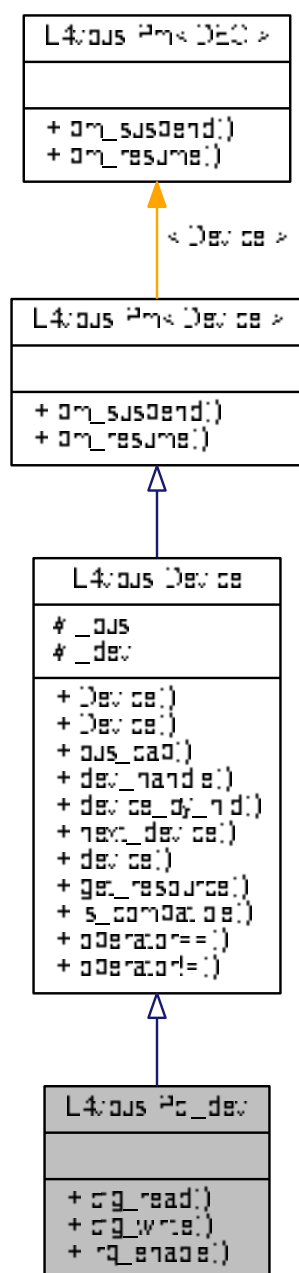
The documentation for this class was generated from the following file:

- I4/vbus/vbus

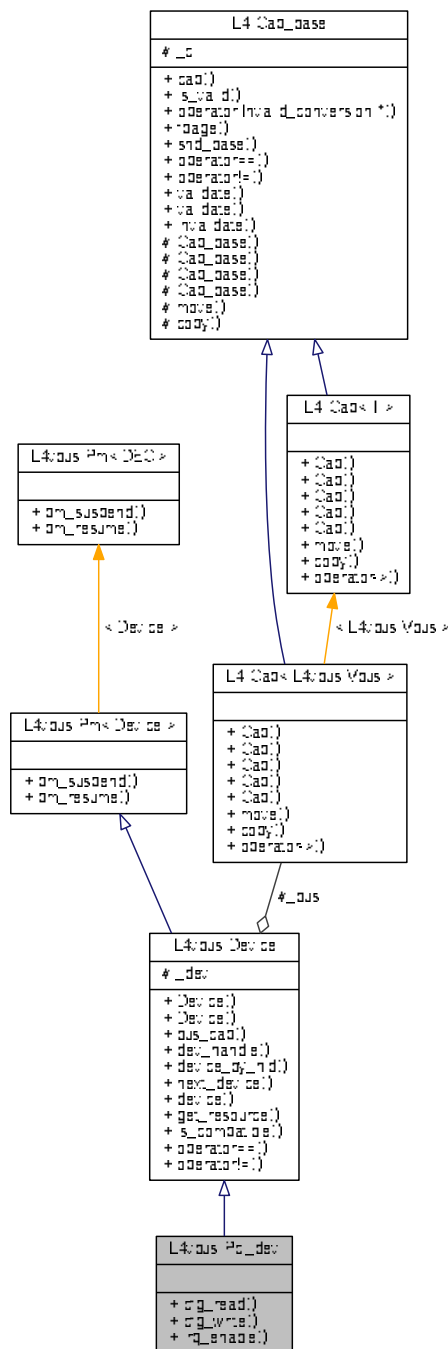
## 14.315 L4vbus::Pci\_dev Class Reference

A PCI device.

Inheritance diagram for L4vbus::Pci\_dev:



Collaboration diagram for L4vbus::Pci\_dev:



## Public Member Functions

- `int cfg_read (l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const`  
*Read from the device's vPCI configuration space.*
- `int cfg_write (l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const`  
*Write to the device's vPCI configuration space.*
- `int irq_enable (unsigned char *trigger, unsigned char *polarity) const`  
*Enable the device's PCI interrupt.*

## Additional Inherited Members

### 14.315.1 Detailed Description

A PCI device.

Definition at line 87 of file [vbus\\_pci](#).

### 14.315.2 Member Function Documentation

#### 14.315.2.1 `cfg_read()`

```
int L4vbus::Pci_dev::cfg_read (
 14_uint32_t reg,
 14_uint32_t * value,
 14_uint32_t width) const [inline]
```

Read from the device's vPCI configuration space.

#### Parameters

|     |              |                                         |
|-----|--------------|-----------------------------------------|
|     | <i>reg</i>   | Register in configuration space to read |
| out | <i>value</i> | Value that has been read                |
|     | <i>width</i> | Width to read in bits (e.g. 8, 16, 32)  |

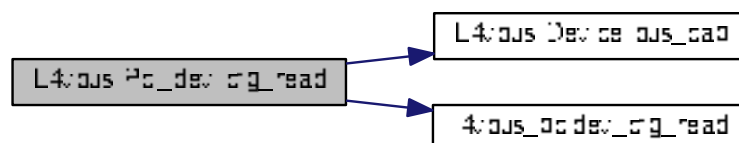
#### Returns

0 on success, else failure

Definition at line 99 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pcidev\\_cfg\\_read\(\)](#).

Here is the call graph for this function:



14.315.2.2 `cfg_write()`

```
int L4vbus::Pci_dev::cfg_write (
 l4_uint32_t reg,
 l4_uint32_t value,
 l4_uint32_t width) const [inline]
```

Write to the device's vPCI configuration space.

## Parameters

|              |                                          |
|--------------|------------------------------------------|
| <i>reg</i>   | Register in configuration space to write |
| <i>value</i> | Value to write                           |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32)  |

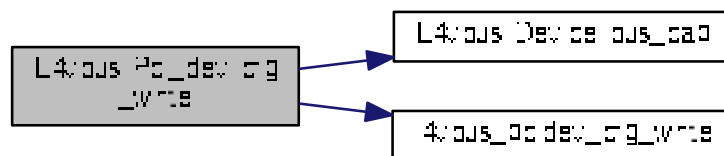
## Returns

0 on success, else failure

Definition at line 115 of file `vbus_pci`.

References `L4vbus::Device::_dev`, `L4vbus::Device::bus_cap()`, and `l4vbus_pcidev_cfg_write()`.

Here is the call graph for this function:

14.315.2.3 `irq_enable()`

```
int L4vbus::Pci_dev::irq_enable (
 unsigned char * trigger,
 unsigned char * polarity) const [inline]
```

Enable the device's PCI interrupt.

## Parameters

|     |                 |                                       |
|-----|-----------------|---------------------------------------|
| out | <i>trigger</i>  | False if interrupt is level-triggered |
| out | <i>polarity</i> | True if interrupt is of low polarity  |

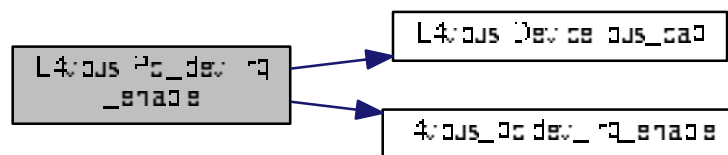
**Returns**

On success: Interrupt line to be used, else failure

Definition at line 131 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pcidev\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



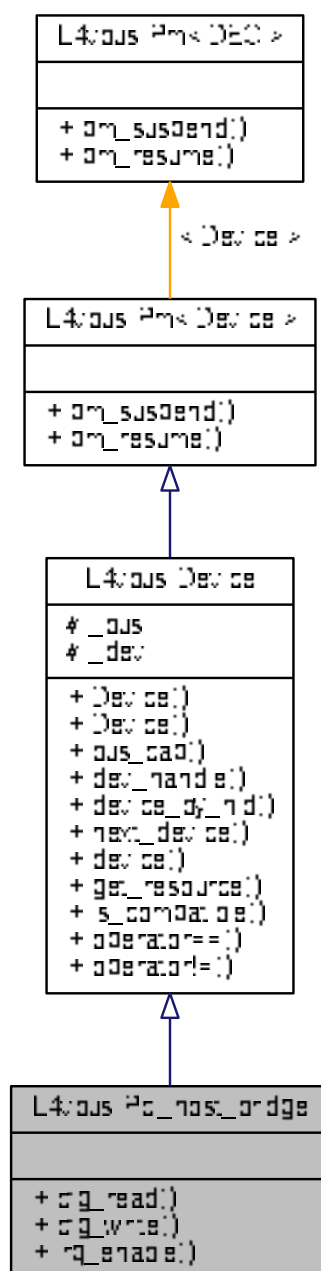
The documentation for this class was generated from the following file:

- [l4/vbus/vbus\\_pci](#)

## 14.316 L4vbus::Pci\_host\_bridge Class Reference

A Pci host bridge.

Inheritance diagram for L4vbus::Pci\_host\_bridge:





- Generated for L4Re by Doxygen



- int [irq\\_enable](#) ([l4\\_uint32\\_t](#) bus, [l4\\_uint32\\_t](#) devfn, int pin, unsigned char \*trigger, unsigned char \*polarity) const

*Enable PCI interrupt for a specific device using the PCI root bridge.*

## Additional Inherited Members

### 14.316.1 Detailed Description

A Pci host bridge.

Definition at line 20 of file [vbus\\_pci](#).

### 14.316.2 Member Function Documentation

#### 14.316.2.1 [cfg\\_read\(\)](#)

```
int L4vbus::Pci_host_bridge::cfg_read (
 l4_uint32_t bus,
 l4_uint32_t devfn,
 l4_uint32_t reg,
 l4_uint32_t * value,
 l4_uint32_t width) const [inline]
```

Read from the vPCI configuration space using the PCI root bridge.

#### Parameters

|     |              |                                                                    |
|-----|--------------|--------------------------------------------------------------------|
|     | <i>bus</i>   | Bus number                                                         |
|     | <i>devfn</i> | <a href="#">Device</a> id (upper 16bit) and function (lower 16bit) |
|     | <i>reg</i>   | Register in configuration space to read                            |
| out | <i>value</i> | Value that has been read                                           |
|     | <i>width</i> | Width to read in bits (e.g. 8, 16, 32)                             |

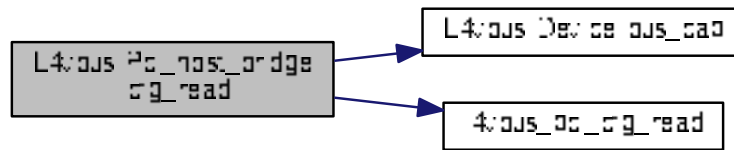
#### Returns

0 on success, else failure

Definition at line 34 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_read\(\)](#).

Here is the call graph for this function:



#### 14.316.2.2 `cfg_write()`

```

int L4vbus::Pci_host_bridge::cfg_write (
 l4_uint32_t bus,
 l4_uint32_t devfn,
 l4_uint32_t reg,
 l4_uint32_t value,
 l4_uint32_t width) const [inline]

```

Write to the vPCI configuration space using the PCI root bridge.

##### Parameters

|              |                                                                    |
|--------------|--------------------------------------------------------------------|
| <i>bus</i>   | Bus number                                                         |
| <i>devfn</i> | <a href="#">Device</a> id (upper 16bit) and function (lower 16bit) |
| <i>reg</i>   | Register in configuration space to write                           |
| <i>value</i> | Value to write                                                     |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32)                            |

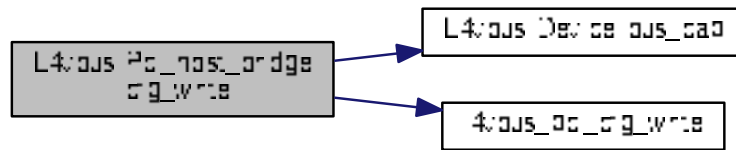
##### Returns

0 on success, else failure

Definition at line 53 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_write\(\)](#).

Here is the call graph for this function:



### 14.316.2.3 irq\_enable()

```

int L4vbus::Pci_host_bridge::irq_enable (
 l4_uint32_t bus,
 l4_uint32_t devfn,
 int pin,
 unsigned char * trigger,
 unsigned char * polarity) const [inline]

```

Enable PCI interrupt for a specific device using the PCI root bridge.

#### Parameters

|     |                 |                                                                     |
|-----|-----------------|---------------------------------------------------------------------|
|     | <i>bus</i>      | Bus number                                                          |
|     | <i>devfn</i>    | Device id (upper 16bit) and function (lower 16bit)                  |
|     | <i>pin</i>      | Interrupt pin (normally as reported in configuration register INTR) |
| out | <i>trigger</i>  | False if interrupt is level-triggered                               |
| out | <i>polarity</i> | True if interrupt is of low polarity                                |

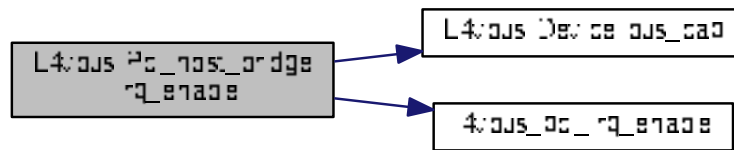
#### Returns

On success: Interrupt line to be used, else failure

Definition at line 74 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



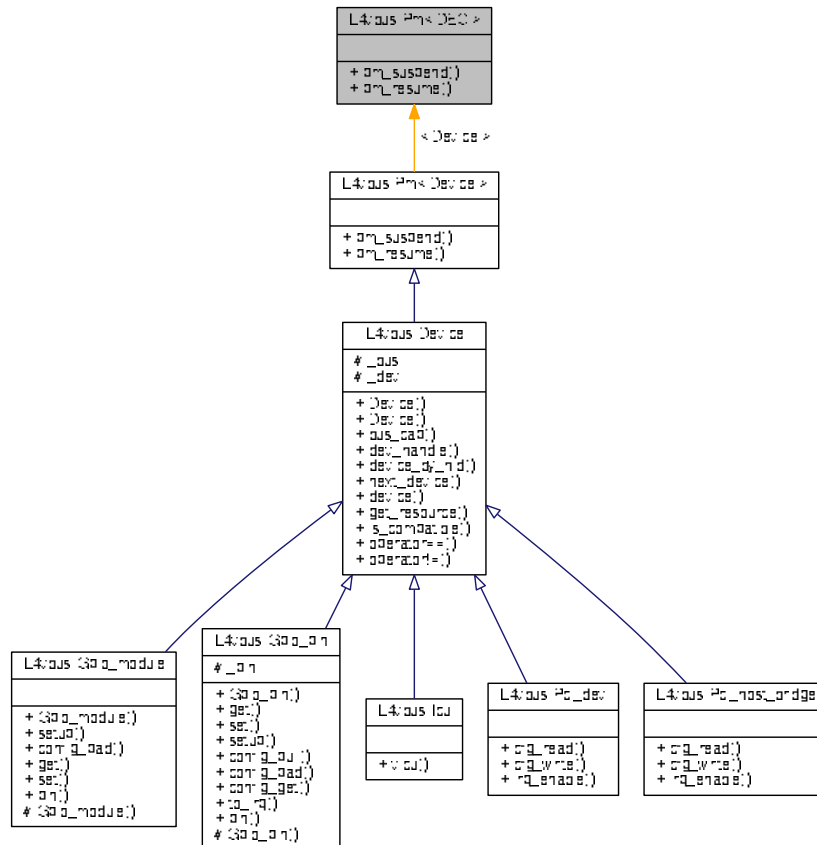
The documentation for this class was generated from the following file:

- l4vbus/vbus\_pci

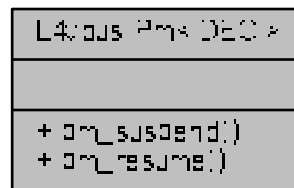
## 14.317 L4vbus::Pm< DEC > Class Template Reference

Power-management API mixin.

Inheritance diagram for L4vbus::Pm< DEC >:



Collaboration diagram for L4vbus::Pm< DEC >:



## Public Member Functions

- `int pm_suspend () const`  
*Suspend the module.*
- `int pm_resume () const`  
*Resume the module.*

### 14.317.1 Detailed Description

```
template<typename DEC>
class L4vbus::Pm< DEC >
```

Power-management API mixin.

Definition at line 47 of file [vbus](#).

The documentation for this class was generated from the following file:

- `I4/vbus/vbus`

## 14.318 L4vbus::Vbus Class Reference

The virtual bus ([Vbus](#)) interface.





- Generated for L4Re by Doxygen

*Get the root device of the device tree of this bus.*

- int [assign\\_dma\\_domain](#) (unsigned domain\_id, unsigned flags, [L4::Cap](#)< [L4Re::Dma\\_space](#) > dma\_space) const

*Assign an [L4Re::Dma\\_space](#) to a DMA domain.*

- int [assign\\_dma\\_domain](#) (unsigned domain\_id, unsigned flags, [L4::Cap](#)< [L4::Task](#) > dma\_space) const

*Assign a kernel DMA space to a DMA domain.*

## Additional Inherited Members

### 14.318.1 Detailed Description

The virtual bus ([Vbus](#)) interface.

See also

[L4Re Vbus API](#)

Definition at line 253 of file [vbus](#).

### 14.318.2 Member Function Documentation

#### 14.318.2.1 [assign\\_dma\\_domain\(\)](#) [1/2]

```
int L4vbus::Vbus::assign_dma_domain (
 unsigned domain_id,
 unsigned flags,
 L4::Cap< L4Re::Dma_space > dma_space) const [inline]
```

Assign an [L4Re::Dma\\_space](#) to a DMA domain.

#### Parameters

|                  |                                                                                                                                |
|------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <i>domain_id</i> | DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used. |
| <i>flags</i>     | A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .                                                              |
| <i>dma_space</i> | The DMA space capability to bind or unbind, this must be an <a href="#">L4Re::Dma_space</a>                                    |

#### Return values

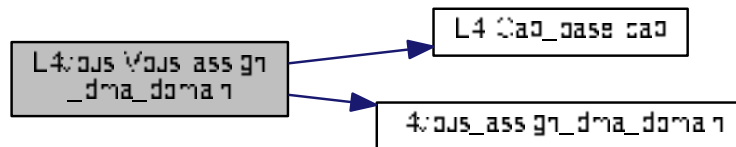
|                   |                                                                                 |
|-------------------|---------------------------------------------------------------------------------|
| <i>0</i>          | Operation completed successfully.                                               |
| <i>-L4_ENOENT</i> | The vbus does not support a global DMA domain or no DMA domain could be found.  |
| <i>-L4_EINVAL</i> | Invalid argument used.                                                          |
| <i>-L4_EBUSY</i>  | DMA domain is already active, this means another DMA space is already assigned. |



Definition at line 303 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#), and [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#).

Here is the call graph for this function:



#### 14.318.2.2 assign\_dma\_domain() [2/2]

```
int L4vbus::Vbus::assign_dma_domain (
 unsigned domain_id,
 unsigned flags,
 L4::Cap< L4::Task > dma_space) const [inline]
```

Assign a kernel DMA space to a DMA domain.

##### Parameters

|                  |                                                                                                                                         |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <i>domain_id</i> | DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.          |
| <i>flags</i>     | A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .                                                                       |
| <i>dma_space</i> | The DMA space capability to bind or unbind, this must be a kernel DMA space ( <a href="#">L4::Task</a> created with L4_PROTO_DMA_SPACE) |

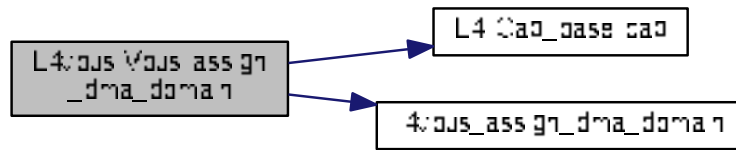
##### Return values

|            |                                                                                 |
|------------|---------------------------------------------------------------------------------|
| 0          | Operation completed successfully.                                               |
| -L4_ENOENT | The vbus does not support a global DMA domain or no DMA domain could be found.  |
| -L4_EINVAL | Invalid argument used.                                                          |
| -L4_EBUSY  | DMA domain is already active, this means another DMA space is already assigned. |

Definition at line 328 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#), and [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#).

Here is the call graph for this function:



### 14.318.2.3 release\_resource()

```
int L4vbus::Vbus::release_resource (
 l4vbus_resource_t * res) const [inline]
```

Release the given resource from the bus.

#### Parameters

|            |                                                    |
|------------|----------------------------------------------------|
| <i>res</i> | The resource that shall be requested from the bus. |
|------------|----------------------------------------------------|

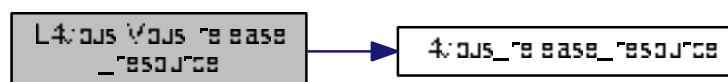
#### Returns

>=0 on success, <0 on error.

Definition at line 273 of file [vbus](#).

References [l4vbus\\_release\\_resource\(\)](#).

Here is the call graph for this function:



## 14.318.2.4 request\_resource()

```
int L4vbus::Vbus::request_resource (
 l4vbus_resource_t * res,
 int flags = 0) const [inline]
```

Request the given resource from the bus.

## Parameters

|              |                                                    |
|--------------|----------------------------------------------------|
| <i>res</i>   | The resource that shall be requested from the bus. |
| <i>flags</i> | The flags for the request.                         |

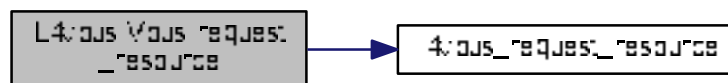
## Returns

>=0 on success, <0 on error.

Definition at line 263 of file [vbus](#).

References [l4vbus\\_request\\_resource\(\)](#).

Here is the call graph for this function:



## 14.318.2.5 root()

```
Device L4vbus::Vbus::root () const [inline]
```

Get the root device of the device tree of this bus.

## Returns

A [Vbus](#) device representing the root of the device tree.

Definition at line 282 of file [vbus](#).

References [L4VBUS\\_ROOT\\_BUS](#).

The documentation for this class was generated from the following file:

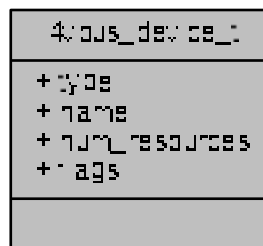
- [l4/vbus/vbus](#)

## 14.319 l4vbus\_device\_t Struct Reference

Detailed information about a vbus device.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_device\_t:



### Data Fields

- [l4\\_uint32\\_t](#) `type`  
*Bitfield of supported sub-interfaces, see [l4vbus\\_iface\\_type\\_t](#).*
- `char` [name](#) [L4VBUS\_DEV\_NAME\_LEN]  
*Name.*
- unsigned [num\\_resources](#)  
*Number of resources for this device.*
- unsigned [flags](#)  
*Flags, see [l4vbus\\_device\\_flags\\_t](#).*

### 14.319.1 Detailed Description

Detailed information about a vbus device.

Definition at line 56 of file [vbus\\_types.h](#).

The documentation for this struct was generated from the following file:

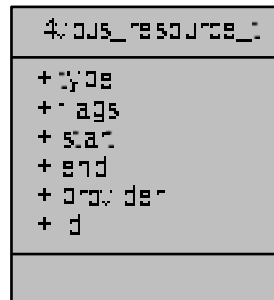
- [l4/vbus/vbus\\_types.h](#)

## 14.320 l4vbus\_resource\_t Struct Reference

Description of a single vbus resource.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_resource\_t:



### Data Fields

- [l4\\_uint16\\_t type](#)  
*Resource type, see [l4vbus\\_resource\\_type\\_t](#).*
- [l4\\_uint16\\_t flags](#)  
*Flags.*
- [l4vbus\\_paddr\\_t start](#)  
*Start of resource range.*
- [l4vbus\\_paddr\\_t end](#)  
*End of resource range (inclusive)*
- [l4vbus\\_device\\_handle\\_t provider](#)  
*Device handle of the provider of the resource.*
- [l4\\_uint32\\_t id](#)  
*Resource ID (4 bytes), usually a 4 letter ASCII name is used.*

### 14.320.1 Detailed Description

Description of a single vbus resource.

Definition at line 23 of file [vbus\\_types.h](#).

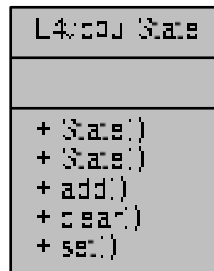
The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_types.h](#)

## 14.321 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

Collaboration diagram for L4vcpu::State:



### Public Member Functions

- [State](#) (unsigned v)  
*Initialize state.*
- void [add](#) (unsigned bits) throw ()  
*Add flags.*
- void [clear](#) (unsigned bits) throw ()  
*Clear flags.*
- void [set](#) (unsigned v) throw ()  
*Set flags.*

### 14.321.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line 31 of file [vcpu](#).

### 14.321.2 Constructor & Destructor Documentation

#### 14.321.2.1 State()

```
L4vcpu::State::State (
 unsigned v) [inline], [explicit]
```

Initialize state.

## Parameters

|          |                |
|----------|----------------|
| <i>v</i> | Initial state. |
|----------|----------------|

Definition at line 41 of file [vcpu](#).

### 14.321.3 Member Function Documentation

#### 14.321.3.1 add()

```
void L4vcpu::State::add (
 unsigned bits) throw () [inline]
```

Add flags.

## Parameters

|             |                          |
|-------------|--------------------------|
| <i>bits</i> | Bits to add to the word. |
|-------------|--------------------------|

Definition at line 48 of file [vcpu](#).

#### 14.321.3.2 clear()

```
void L4vcpu::State::clear (
 unsigned bits) throw () [inline]
```

Clear flags.

## Parameters

|             |                            |
|-------------|----------------------------|
| <i>bits</i> | Bits to clear in the word. |
|-------------|----------------------------|

Definition at line 55 of file [vcpu](#).

#### 14.321.3.3 set()

```
void L4vcpu::State::set (
 unsigned v) throw () [inline]
```

Set flags.

**Parameters**

|                |                                               |
|----------------|-----------------------------------------------|
| <code>v</code> | Set the word to the value of <code>v</code> . |
|----------------|-----------------------------------------------|

Definition at line [62](#) of file [vcpu](#).

The documentation for this class was generated from the following file:

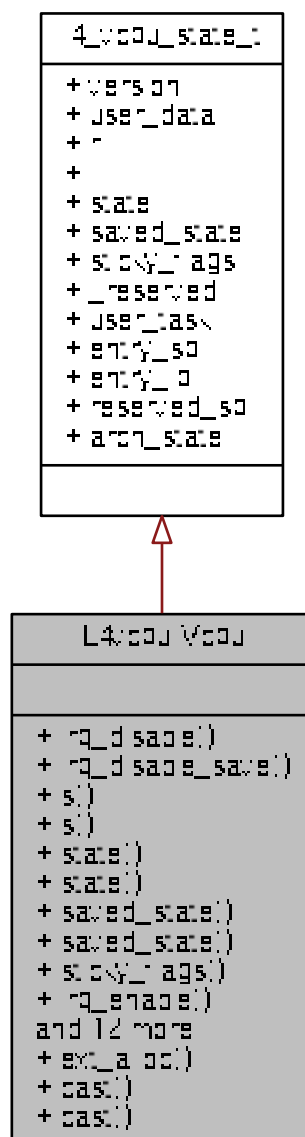
- `l4/vcpu/vcpu`

## 14.322 L4vcpu::Vcpu Class Reference

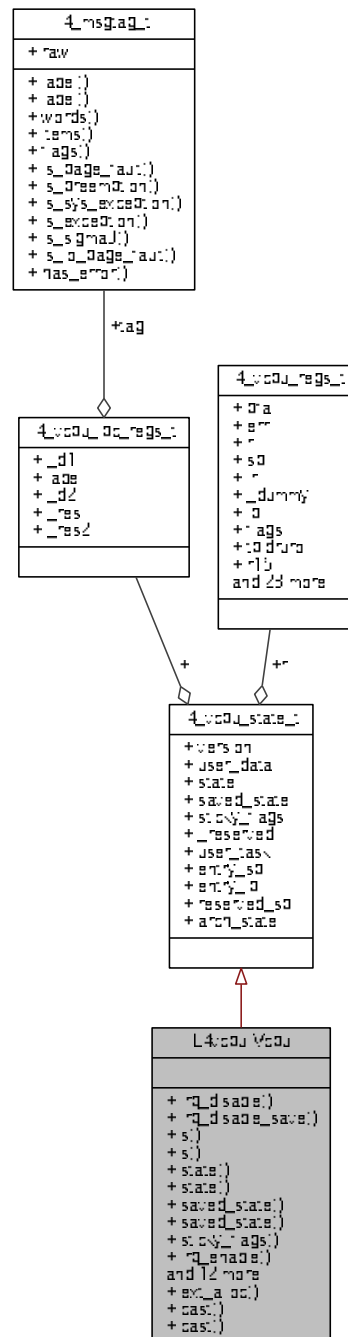
C++ implementation of the vCPU save state area.



Inheritance diagram for L4vcpu::Vcpu:



Collaboration diagram for L4vcpu::Vcpu:



## Public Member Functions

- `void irq_disable () throw ()`  
Disable the vCPU for event delivery.
- `unsigned irq_disable_save () throw ()`  
Disable the vCPU for event delivery and return previous state.
- `State * state () throw ()`

- Get state word.*
- [State state](#) () const throw ()
- Get state word.*
- [State \\* saved\\_state](#) () throw ()
- Get saved\_state word.*
- [State saved\\_state](#) () const throw ()
- Get saved\_state word.*
- [l4\\_uint16\\_t sticky\\_flags](#) () const throw ()
- Get sticky flags.*
- void [irq\\_enable](#) ([l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) throw ()
- Enable the vCPU for event delivery.*
- void [irq\\_restore](#) (unsigned s, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) throw ()
- Restore a previously saved IRQ/event state.*
- void [wait\\_for\\_event](#) ([l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) throw ()
- Wait for event.*
- void [task](#) ([L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid) throw ()
- Set the task of the vCPU.*
- int [is\\_page\\_fault\\_entry](#) () const
- Return whether the entry reason was a page fault.*
- int [is\\_irq\\_entry](#) () const
- Return whether the entry reason was an IRQ/IPC message.*
- [l4\\_vcpu\\_regs\\_t](#) \* [r](#) () throw ()
- Return pointer to register state.*
- [l4\\_vcpu\\_regs\\_t](#) const \* [r](#) () const throw ()
- Return pointer to register state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t](#) \* [i](#) () throw ()
- Return pointer to IPC state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t](#) const \* [i](#) () const throw ()
- Return pointer to IPC state.*
- void [entry\\_sp](#) ([l4\\_umword\\_t](#) sp)
- Set vCPU entry stack pointer.*
- void [entry\\_ip](#) ([l4\\_umword\\_t](#) ip)
- Set vCPU entry instruction pointer.*
- void [print\\_state](#) (const char \*prefix="") const throw ()
- Print the state of the vCPU.*

## Static Public Member Functions

- static int [ext\\_alloc](#) ([Vcpu](#) \*\*vcpu, [l4\\_addr\\_t](#) \*ext\_state, [L4::Cap](#)< [L4::Task](#) > task=[L4Re::Env::env](#)() ->task(), [L4::Cap](#)< [L4Re::Rm](#) > rm=[L4Re::Env::env](#)() ->rm()) throw ()
- Allocate state area for an extended vCPU.*
- static [Vcpu](#) \* [cast](#) (void \*x) throw ()
- Cast a void pointer to a class pointer.*
- static [Vcpu](#) \* [cast](#) ([l4\\_addr\\_t](#) x) throw ()
- Cast an address to a class pointer.*

### 14.322.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 72 of file [vcpu](#).

### 14.322.2 Member Function Documentation

#### 14.322.2.1 `cast()` [1/2]

```
static Vcpu* L4vcpu::Vcpu::cast (
 void * x) throw) [inline], [static]
```

Cast a void pointer to a class pointer.

##### Parameters

|                |          |
|----------------|----------|
| <code>x</code> | Pointer. |
|----------------|----------|

##### Returns

Pointer to [Vcpu](#) class.

Definition at line 265 of file [vcpu](#).

#### 14.322.2.2 `cast()` [2/2]

```
static Vcpu* L4vcpu::Vcpu::cast (
 l4_addr_t x) throw) [inline], [static]
```

Cast an address to a class pointer.

##### Parameters

|                |          |
|----------------|----------|
| <code>x</code> | Pointer. |
|----------------|----------|

##### Returns

Pointer to [Vcpu](#) class.

Definition at line 275 of file [vcpu](#).

## 14.322.2.3 entry\_ip()

```
void L4vcpu::Vcpu::entry_ip (
 l4_umword_t ip) [inline]
```

Set vCPU entry instruction pointer.

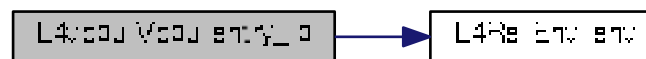
## Parameters

|           |                                     |
|-----------|-------------------------------------|
| <i>ip</i> | Instruction pointer address to set. |
|-----------|-------------------------------------|

Definition at line 239 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::entry\\_ip](#), [L4Re::Env::env\(\)](#), and [L4\\_CV](#).

Here is the call graph for this function:



## 14.322.2.4 entry\_sp()

```
void L4vcpu::Vcpu::entry_sp (
 l4_umword_t sp) [inline]
```

Set vCPU entry stack pointer.

## Parameters

|           |                               |
|-----------|-------------------------------|
| <i>sp</i> | Stack pointer address to set. |
|-----------|-------------------------------|

## Note

The value is only used when entering from a user-task.

Definition at line 232 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::entry\\_sp](#).

14.322.2.5 `ext_alloc()`

```
static int L4vcpu::Vcpu::ext_alloc (
 Vcpu ** vcpu,
 l4_addr_t * ext_state,
 L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
 L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm()) throw () [static]
```

Allocate state area for an extended vCPU.

## Return values

|                        |                                     |
|------------------------|-------------------------------------|
| <code>vcpu</code>      | Allocated vcpu-state area.          |
| <code>ext_state</code> | Allocated extended vcpu-state area. |

## Parameters

|                   |                                                                           |
|-------------------|---------------------------------------------------------------------------|
| <code>task</code> | Task to use for allocation, defaults to own task.                         |
| <code>rm</code>   | Region manager to use for allocation defaults to standard region manager. |

## Returns

0 for success, error code otherwise

14.322.2.6 `i()` [1/2]

```
l4_vcpu_ipc_regs_t* L4vcpu::Vcpu::i () throw () [inline]
```

Return pointer to IPC state.

## Returns

Pointer to IPC state.

Definition at line 216 of file `vcpu`.

References `l4_vcpu_state_t::i`.

14.322.2.7 `i()` [2/2]

```
l4_vcpu_ipc_regs_t const* L4vcpu::Vcpu::i () const throw () [inline]
```

Return pointer to IPC state.

## Returns

Pointer to IPC state.

Definition at line 223 of file `vcpu`.

References `l4_vcpu_state_t::i`.

## 14.322.2.8 irq\_disable\_save()

```
unsigned L4vcpu::Vcpu::irq_disable_save () throw () [inline]
```

Disable the vCPU for event delivery and return previous state.

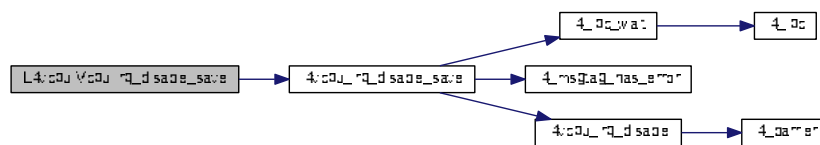
## Returns

IRQ state before disabling IRQs.

Definition at line 85 of file [vcpu](#).

References [l4vcpu\\_irq\\_disable\\_save\(\)](#).

Here is the call graph for this function:



## 14.322.2.9 irq\_enable()

```
void L4vcpu::Vcpu::irq_enable (
 l4_utcb_t * utcb,
 l4vcpu_event_hdl_t do_event_work_cb,
 l4vcpu_setup_ipc_t setup_ipc) throw () [inline]
```

Enable the vCPU for event delivery.

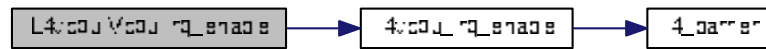
## Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <i>utcb</i>             | The UTCB to use.                                                                                           |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending.                      |
| <i>setup_ipc</i>        | Call-back function that is called before an IPC operation is called, and before event delivery is enabled. |

Definition at line 142 of file [vcpu](#).

References [l4vcpu\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



#### 14.322.2.10 irq\_restore()

```

void L4vcpu::Vcpu::irq_restore (
 unsigned s,
 l4_utcb_t * utcb,
 l4vcpu_event_hndl_t do_event_work_cb,
 l4vcpu_setup_ipc_t setup_ipc) throw () [inline]

```

Restore a previously saved IRQ/event state.

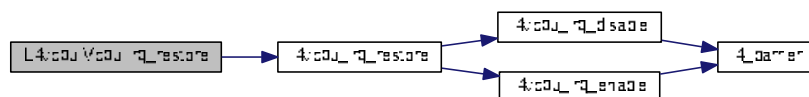
##### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <i>s</i>                | IRQ state to be restored.                                                                                  |
| <i>utcb</i>             | The UTCB to use.                                                                                           |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending.                      |
| <i>setup_ipc</i>        | Call-back function that is called before an IPC operation is called, and before event delivery is enabled. |

Definition at line 157 of file [vcpu](#).

References [l4\\_vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



#### 14.322.2.11 is\_irq\_entry()

```

int L4vcpu::Vcpu::is_irq_entry () const [inline]

```



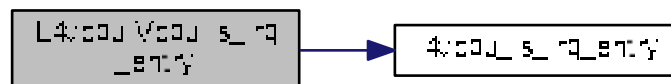
Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 195 of file [vcpu](#).

References [l4vcpu\\_is\\_irq\\_entry\(\)](#).

Here is the call graph for this function:



#### 14.322.2.12 is\_page\_fault\_entry()

```
int L4vcpu::Vcpu::is_page_fault_entry () const [inline]
```

Return whether the entry reason was a page fault.

return 0 if not, !=0 otherwise.

Definition at line 188 of file [vcpu](#).

References [l4vcpu\\_is\\_page\\_fault\\_entry\(\)](#).

Here is the call graph for this function:



**14.322.2.13** `r()` [1/2]

```
l4_vcpu_regs_t* L4vcpu::Vcpu::r () throw () [inline]
```

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 202 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**14.322.2.14** `r()` [2/2]

```
l4_vcpu_regs_t const* L4vcpu::Vcpu::r () const throw () [inline]
```

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 209 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**14.322.2.15** `saved_state()` [1/2]

```
State* L4vcpu::Vcpu::saved_state () throw () [inline]
```

Get `saved_state` word.

**Returns**

Pointer to `saved_state` word in the vCPU

Definition at line 113 of file `vcpu`.

**14.322.2.16 saved\_state()** [2/2]

```
State L4vcpu::Vcpu::saved_state () const throw () [inline]
```

Get saved\_state word.

**Returns**

Pointer to saved\_state word in the vCPU

Definition at line 123 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::saved\\_state](#).

**14.322.2.17 state()** [1/2]

```
State* L4vcpu::Vcpu::state () throw () [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 95 of file [vcpu](#).

**14.322.2.18 state()** [2/2]

```
State L4vcpu::Vcpu::state () const throw () [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 106 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::state](#).

**14.322.2.19 task()**

```
void L4vcpu::Vcpu::task (
 L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid) throw () [inline]
```

Set the task of the vCPU.

## Parameters

|             |                                        |
|-------------|----------------------------------------|
| <i>task</i> | Task to set, defaults to invalid task. |
|-------------|----------------------------------------|

Definition at line 181 of file [vcpu](#).

## 14.322.2.20 wait\_for\_event()

```
void L4vcpu::Vcpu::wait_for_event (
 l4_utcb_t * utcb,
 l4vcpu_event_hndl_t do_event_work_cb,
 l4vcpu_setup_ipc_t setup_ipc) throw () [inline]
```

Wait for event.

## Parameters

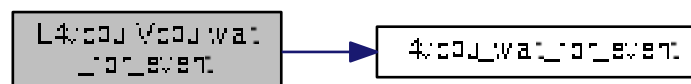
|                         |                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------|
| <i>utcb</i>             | The UTCB to use.                                                                      |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending. |
| <i>setup_ipc</i>        | Call-back function that is called before an IPC operation is called.                  |

Note that event delivery remains disabled after this function returns.

Definition at line 173 of file [vcpu](#).

References [l4vcpu\\_wait\\_for\\_event\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- `l4/vcpu/vcpu`





- long `register_ds` (`L4::lpc::Cap< L4Re::Dataspace >` ds\_cap, `l4_uint64_t` base, `l4_umword_t` offset, `l4_umword_t` size)  
*Register a shared data space with VIRTIO host.*
- long `register_iface` (`L4::lpc::Cap< L4::Triggerable >` guest\_irq, `L4::lpc::Out< L4::Cap< L4::Triggerable > >` host\_irq, `L4::lpc::Out< L4::Cap< L4Re::Dataspace > >` config\_ds)  
*Register client to the L4-VIRTIO device.*
- long `device_config` (`L4::lpc::Out< L4::Cap< L4Re::Dataspace > >` config\_ds, `l4_addr_t` \*ds\_offset)  
*Get the dataspace with the L4virtio configuration page.*
- long `device_notification_irq` (unsigned index, `L4::lpc::Out< L4::Cap< L4::Triggerable > >` irq)  
*Get the notification interrupt corresponding to the given index.*

## Additional Inherited Members

### 14.323.1 Detailed Description

IPC interface for virtio over L4 IPC.

The L4virtio protocol is an adaption of the mmio virtio transport 1.0(4). This interface allows to exchange the necessary resources: device configuration page, notification interrupts and dataspace for payload.

Notification interrupts can be configured independently for changes to the configuration space and each queue through special L4virtio-specific `notify_index` fields in the config page and queue configuration. The interface distinguishes between device-to-driver and driver-to-device notification interrupts.

Device-to-driver interrupts are configured via the ICU interface. The device announces the maximum number of supported interrupts via `lcu::info()`. The driver can then bind interrupts using `lcu::bind()`.

Driver-to-device interrupts must be requested from the device through `device_notification_irq()`.

Definition at line 50 of file `l4virtio`.

### 14.323.2 Member Function Documentation

#### 14.323.2.1 `config_queue()`

```
long L4virtio::Device::config_queue (
 unsigned queue)
```

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

#### Parameters

|                    |                                             |
|--------------------|---------------------------------------------|
| <code>queue</code> | Queue index for the queue to be configured. |
|--------------------|---------------------------------------------|

## Return values

|            |                                               |
|------------|-----------------------------------------------|
| 0          | on success.                                   |
| -L4_EIO    | The queue's status is invalid.                |
| -L4_ERANGE | The queue index exceeds the number of queues. |
| -L4_EINVAL | Otherwise.                                    |

## 14.323.2.2 device\_config()

```
long L4virtio::Device::device_config (
 L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds,
 l4_addr_t * ds_offset)
```

Get the dataspace with the [L4virtio](#) configuration page.

## Parameters

|                  |                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------|
| <i>config_ds</i> | Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space. |
| <i>ds_offset</i> | Offset into the dataspace where the device configuration structure starts.                    |

## 14.323.2.3 device\_notification\_irq()

```
long L4virtio::Device::device_notification_irq (
 unsigned index,
 L4::Ipc::Out< L4::Cap< L4::Triggerable > > irq)
```

Get the notification interrupt corresponding to the given index.

## Parameters

|     |              |                                  |
|-----|--------------|----------------------------------|
|     | <i>index</i> | Index of the interrupt.          |
| out | <i>irq</i>   | Triggerable for the given index. |

## Return values

|           |                                           |
|-----------|-------------------------------------------|
| L4_EOK    | Success.                                  |
| L4_ENOSYS | IRQ notification not supported by device. |
| <0        | Other error.                              |

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.



## 14.323.2.4 register\_ds()

```
long L4virtio::Device::register_ds (
 L4::Ipc::Cap< L4Re::Dataspace > ds_cap,
 l4_uint64_t base,
 l4_umword_t offset,
 l4_umword_t size)
```

Register a shared data space with VIRTIO host.

## Parameters

|               |                                                                                                                                                                                     |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ds_cap</i> | Data-space capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host. |
| <i>base</i>   | VIRTIO guest physical start address of shared memory region                                                                                                                         |
| <i>offset</i> | Offset within the data space that is attached to the given <i>base</i> in the guest physical memory.                                                                                |
| <i>size</i>   | Size of the memory region in the guest                                                                                                                                              |

## Returns

0 on success, < 0 on error

## 14.323.2.5 register\_iface()

```
long L4virtio::Device::register_iface (
 L4::Ipc::Cap< L4::Triggerable > guest_irq,
 L4::Ipc::Out< L4::Cap< L4::Triggerable > > host_irq,
 L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds)
```

Register client to the L4-VIRTIO device.

## Parameters

|                  |                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------|
| <i>guest_irq</i> | IRQ capability for valid IRQ object for device-to-driver notifications.                       |
| <i>host_irq</i>  | Capability selector for receiving the driver-to-device notifications IRQ capability.          |
| <i>config_ds</i> | Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space. |

## Return values

|                   |                                                                                                         |
|-------------------|---------------------------------------------------------------------------------------------------------|
| <i>L4_EOK</i>     | Success.                                                                                                |
| <i>-L4_ENOSYS</i> | This interface is no longer supported by the server. Use <code>get_device_config()</code> etc. instead. |

**Deprecated** Use `device_config()`, `device_notification_irq()` and `lcu::bind()` instead.

#### 14.323.2.6 set\_status()

```
long L4virtio::Device::set_status (
 unsigned status)
```

Write the VIRTIO status register.

##### Parameters

|               |                                            |
|---------------|--------------------------------------------|
| <i>status</i> | Status word to write to the VIRTIO status. |
|---------------|--------------------------------------------|

##### Returns

0 on success, <0 on error.

##### Note

All other registers are accessed via shared memory.

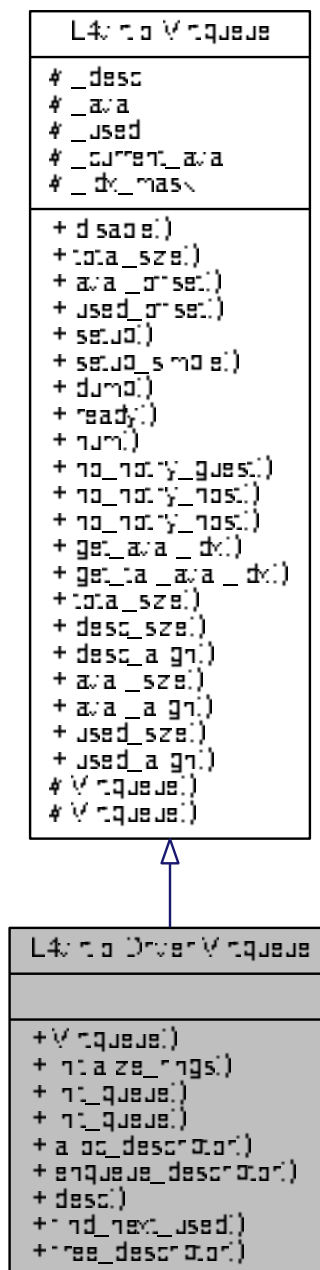
The documentation for this class was generated from the following file:

- l4/l4virtio/l4virtio

## 14.324 L4virtio::Driver::Virtqueue Class Reference

Driver-side implementation of a [Virtqueue](#).

Inheritance diagram for L4virtio::Driver::Virtqueue:



-

- Initialize this virtqueue.*
- [l4\\_uint16\\_t alloc\\_descriptor](#) ()  
*Allocate and return an unused descriptor from the descriptor table.*
- void [enqueue\\_descriptor](#) ([l4\\_uint16\\_t](#) descno)  
*Enqueue a descriptor in the available ring.*
- [Desc & desc](#) ([l4\\_uint16\\_t](#) descno)  
*Return a reference to a descriptor in the descriptor table.*
- [l4\\_uint16\\_t find\\_next\\_used](#) ([l4\\_uint32\\_t](#) \*len=nullptr)  
*Return the next finished block.*
- void [free\\_descriptor](#) ([l4\\_uint16\\_t](#) head, [l4\\_uint16\\_t](#) tail)  
*Free a chained list of descriptors in the descriptor queue.*

## Additional Inherited Members

### 14.324.1 Detailed Description

Driver-side implementation of a [Virtqueue](#).

Adds function for managing the descriptor list, enqueueing new and dequeueing finished requests.

Definition at line [471](#) of file [virtqueue](#).

### 14.324.2 Member Function Documentation

#### 14.324.2.1 [alloc\\_descriptor\(\)](#)

```
l4_uint16_t L4virtio::Driver::Virtqueue::alloc_descriptor () [inline]
```

Allocate and return an unused descriptor from the descriptor table.

The descriptor will be removed from the free list, the content should be considered undefined. After use, it needs to be freed using [free\\_descriptor\(\)](#).

#### Returns

The index of the reserved descriptor or [Virtqueue::Eoq](#) if no free descriptor is available.

Note: the implementation uses  $(2^{16} - 1)$  as the end of queue marker. That means that the final entry in the queue can not be allocated iff the queue size is  $2^{16}$ .

Definition at line [559](#) of file [virtqueue](#).

#### 14.324.2.2 [desc\(\)](#)

```
Desc& L4virtio::Driver::Virtqueue::desc (
 l4_uint16_t descno) [inline]
```

Return a reference to a descriptor in the descriptor table.

## Parameters

|               |                                                           |
|---------------|-----------------------------------------------------------|
| <i>descno</i> | Index of the descriptor, expected to be in correct range. |
|---------------|-----------------------------------------------------------|

Definition at line 593 of file [virtqueue](#).

#### 14.324.2.3 enqueue\_descriptor()

```
void L4virtio::Driver::Virtqueue::enqueue_descriptor (
 14_uint16_t descno) [inline]
```

Enqueue a descriptor in the available ring.

## Parameters

|               |                                          |
|---------------|------------------------------------------|
| <i>descno</i> | Index of the head descriptor to enqueue. |
|---------------|------------------------------------------|

Definition at line 576 of file [virtqueue](#).

#### 14.324.2.4 find\_next\_used()

```
14_uint16_t L4virtio::Driver::Virtqueue::find_next_used (
 14_uint32_t * len = nullptr) [inline]
```

Return the next finished block.

## Parameters

|            |            |                                                                                                                                                                                |
|------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>len</i> | (optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size. |
|------------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Index of the head or Virtqueue::Eoq if no used element is currently available.

Definition at line 612 of file [virtqueue](#).

#### 14.324.2.5 free\_descriptor()

```
void L4virtio::Driver::Virtqueue::free_descriptor (
 14_uint16_t head,
 14_uint16_t tail) [inline]
```

Free a chained list of descriptors in the descriptor queue.

## Parameters

|             |                                                     |
|-------------|-----------------------------------------------------|
| <i>head</i> | Index of the first element in the descriptor chain. |
| <i>tail</i> | Index of the last element in the descriptor chain.  |

Simply takes the descriptor chain and prepends it to the beginning of the free list. Assumes that the list has been correctly chained.

Definition at line 635 of file [virtqueue](#).

14.324.2.6 `init_queue()` [1/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
 unsigned num,
 void * desc,
 void * avail,
 void * used) [inline]
```

Initialize this virtqueue.

## Parameters

|              |                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <i>num</i>   | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).                           |
| <i>desc</i>  | The address of the descriptor table. (Must be <code>Desc_align</code> aligned and at least <code>desc_size(num)</code> bytes in size.) |
| <i>avail</i> | The address of the available ring. (Must be <code>Avail_align</code> aligned and at least <code>avail_size(num)</code> bytes in size.) |
| <i>used</i>  | The address of the used ring. (Must be <code>Used_align</code> aligned and at least <code>used_size(num)</code> bytes in size.)        |

This function sets up the memory and initializes the freelist.

Definition at line 523 of file [virtqueue](#).

14.324.2.7 `init_queue()` [2/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
 unsigned num,
 void * base) [inline]
```

Initialize this virtqueue.

## Parameters

|             |                                                                                                              |
|-------------|--------------------------------------------------------------------------------------------------------------|
| <i>num</i>  | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
| <i>base</i> | The base address for the queue data structure.                                                               |

This function sets up the memory and initializes the freelist.

Definition at line 538 of file [virtqueue](#).

#### 14.324.2.8 initialize\_rings()

```
void L4virtio::Driver::Virtqueue::initialize_rings (
 unsigned num) [inline]
```

Initialize the descriptor table and the index structures of this queue.

##### Parameters

|            |                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
|------------|--------------------------------------------------------------------------------------------------------------|

##### Precondition

The queue must be set up correctly with [setup\(\)](#) or [setup\\_simple\(\)](#).

Definition at line 495 of file [virtqueue](#).

The documentation for this class was generated from the following file:

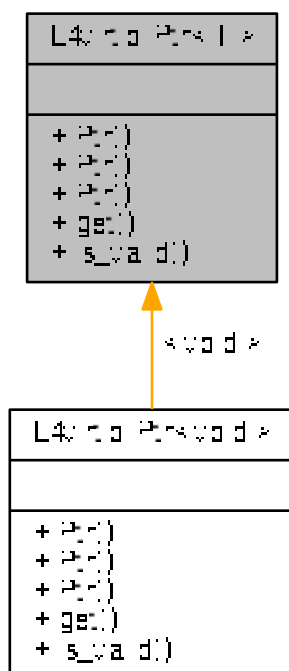
- l4/l4virtio/virtqueue

## 14.325 L4virtio::Ptr< T > Class Template Reference

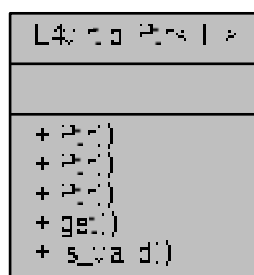
Pointer used in virtio descriptors.



Inheritance diagram for L4virtio::Ptr< T >:



Collaboration diagram for L4virtio::Ptr< T >:



## Public Types

- enum `Invalid_type` { `Invalid` }  
Type for making an invalid (NULL) `Ptr`.

## Public Member Functions

- [Ptr](#) ([Invalid\\_type](#))  
*Make and invalid [Ptr](#).*
- [Ptr](#) ([l4\\_uint64\\_t](#) vm\_addr)  
*Make a [Ptr](#) from a raw 64bit address.*
- [l4\\_uint64\\_t](#) [get](#) () const
- bool [is\\_valid](#) () const

### 14.325.1 Detailed Description

```
template<typename T>
class L4virtio::Ptr< T >
```

Pointer used in virtio descriptors.

As the descriptor contain guest addresses these pointers cannot be dereferenced directly.

Definition at line [56](#) of file [virtqueue](#).

### 14.325.2 Member Enumeration Documentation

#### 14.325.2.1 Invalid\_type

```
template<typename T>
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) [Ptr](#).

#### Enumerator

|         |                                                    |
|---------|----------------------------------------------------|
| Invalid | Use to set a <a href="#">Ptr</a> to invalid (NULL) |
|---------|----------------------------------------------------|

Definition at line [60](#) of file [virtqueue](#).

### 14.325.3 Member Function Documentation

#### 14.325.3.1 get()

```
template<typename T>
l4_uint64_t L4virtio::Ptr< T >::get () const [inline]
```

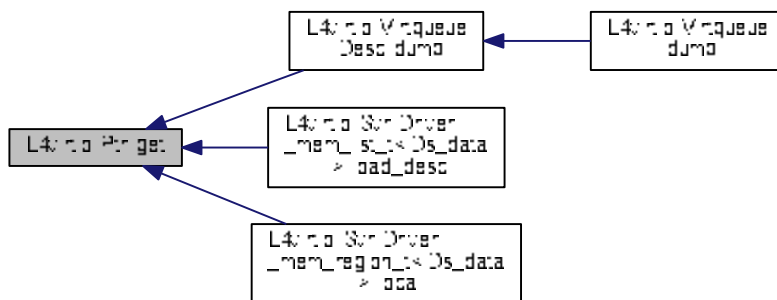
**Returns**

The raw 64bit address of the pointer.

Definition at line 71 of file [virtqueue](#).

Referenced by [L4virtio::Virtqueue::Desc::dump\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< Ds\\_data >::load\\_desc\(\)](#), and [L4virtio::Svr::Driver\\_mem\\_region\\_t< Ds\\_data >::local\(\)](#).

Here is the caller graph for this function:

**14.325.3.2 is\_valid()**

```
template<typename T>
bool L4virtio::Ptr< T >::is_valid () const [inline]
```

**Returns**

true if the pointer is invalid (NULL).

Definition at line 74 of file [virtqueue](#).

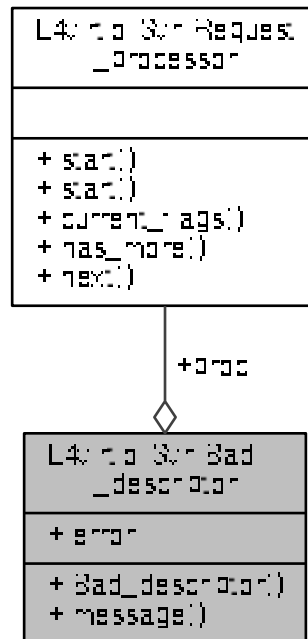
The documentation for this class was generated from the following file:

- `I4/I4virtio/virtqueue`

## 14.326 L4virtio::Svr::Bad\_descriptor Struct Reference

Exception used by Queue to indicate descriptor errors.

Collaboration diagram for L4virtio::Svr::Bad\_descriptor:



### Public Types

- enum [Error](#) {  
[Bad\\_address](#), [Bad\\_rights](#), [Bad\\_flags](#), [Bad\\_next](#),  
[Bad\\_size](#) }

*The error code.*

### Public Member Functions

- [Bad\\_descriptor](#) ([Request\\_processor](#) const \*[proc](#), [Error](#) e)  
*Make a bad descriptor exception.*
- char const \* [message](#) () const  
*Get a human readable description of the error code.*

### Data Fields

- [Request\\_processor](#) const \* [proc](#)  
*The processor that triggered the exception.*

### 14.326.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 332 of file [virtio](#).

### 14.326.2 Member Enumeration Documentation

#### 14.326.2.1 Error

```
enum L4virtio::Svr::Bad_descriptor::Error
```

The error code.

##### Enumerator

|             |                                          |
|-------------|------------------------------------------|
| Bad_address | Address cannot be translated.            |
| Bad_rights  | Missing access rights on memory.         |
| Bad_flags   | Invalid combination of descriptor flags. |
| Bad_next    | Invalid next index.                      |
| Bad_size    | Invalid size of memory block.            |

Definition at line 335 of file [virtio](#).

### 14.326.3 Constructor & Destructor Documentation

#### 14.326.3.1 Bad\_descriptor()

```
L4virtio::Svr::Bad_descriptor::Bad_descriptor (
 Request_processor const * proc,
 Error e) [inline]
```

Make a bad descriptor exception.

##### Parameters

|             |                                             |
|-------------|---------------------------------------------|
| <i>proc</i> | The request processor causing the exception |
| <i>e</i>    | The error code.                             |

Definition at line 355 of file [virtio](#).

#### 14.326.4 Member Function Documentation

##### 14.326.4.1 message()

```
char const* L4virtio::Svr::Bad_descriptor::message () const [inline]
```

Get a human readable description of the error code.

##### Returns

Message describing the error.

Definition at line [364](#) of file [virtio](#).

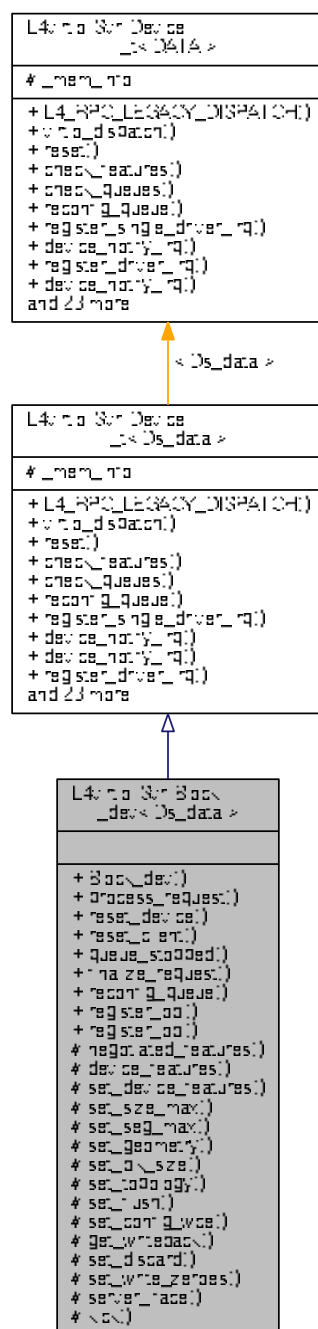
The documentation for this struct was generated from the following file:

- [l4/l4virtio/server/virtio](#)

#### 14.327 L4virtio::Svr::Block\_dev< Ds\_data > Class Template Reference

Base class for virtio block devices.

Inheritance diagram for L4virtio::Svr::Block\_dev< Ds\_data >:







- Reset the actual hardware device.*
- virtual bool `reset_client` ()  
*The client requests reinitialisation of the connection.*
- virtual bool `queue_stopped` ()=0  
*Return true, if the queues should not be processed further.*
- void `finalize_request` (cxx::unique\_ptr< Request > req, unsigned sz, l4\_uint8\_t status=L4VIRTIO\_BLOCK↔\_S\_OK)  
*Releases resources related to a request and notifies the client.*
- int `reconfig_queue` (unsigned idx)  
*callback for client queue-config request*
- L4::Cap< void > `register_obj` (L4::Registry\_iface \*registry, char const \*service=0)  
*Attach device to an object registry.*

## Protected Member Functions

- void `set_size_max` (l4\_uint32\_t sz)  
*Sets the maximum size of any single segment reported to client.*
- void `set_seg_max` (l4\_uint32\_t sz)  
*Sets the maximum number of segments in a request that is reported to client.*
- void `set_geometry` (l4\_uint16\_t cylinders, l4\_uint8\_t heads, l4\_uint8\_t sectors)  
*Set disk geometry that is reported to the client.*
- void `set_blk_size` (l4\_uint32\_t sz)  
*Sets block disk size to be reported to the client.*
- void `set_topology` (l4\_uint8\_t physical\_block\_exp, l4\_uint8\_t alignment\_offset, l4\_uint32\_t min\_io\_size, l4↔\_uint32\_t opt\_io\_size)  
*Sets the I/O alignment information reported back to the client.*
- void `set_flush` ()  
*Enables the flush command.*
- void `set_config_wce` (l4\_uint8\_t writeback)  
*Sets cache mode and enables the the writeback toggle.*
- l4\_uint8\_t `get_writeback` ()  
*Get the writeback field from the configuration space.*
- void `set_discard` (l4\_uint32\_t max\_discard\_sectors, l4\_uint32\_t max\_discard\_seg, l4\_uint32\_t discard↔\_sector\_alignment)  
*Sets constraints for and enables the discard command.*
- void `set_write_zeroes` (l4\_uint32\_t max\_write\_zeroes\_sectors, l4\_uint32\_t max\_write\_zeroes\_seg, l4↔\_uint8\_t write\_zeroes\_may\_unmap)  
*Sets constraints for and enables the write zeroes command.*

## Additional Inherited Members

### 14.327.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_dev< Ds_data >
```

Base class for virtio block devices.

Use this class as a base to implement your own specific block device.

Definition at line 23 of file [virtio-block](#).

## 14.327.2 Constructor & Destructor Documentation

### 14.327.2.1 Block\_dev()

```
template<typename Ds_data>
L4virtio::Svr::Block_dev< Ds_data >::Block_dev (
 l4_uint32_t vendor,
 unsigned queue_size,
 l4_uint64_t capacity,
 bool read_only) [inline]
```

Create a new virtio block device.

#### Parameters

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>vendor</i>     | Vendor ID                                             |
| <i>queue_size</i> | Number of entries to provide in avail and used queue. |
| <i>capacity</i>   | Size of the device in 512-byte sectors.               |
| <i>read_only</i>  | True, if the device should not be writable.           |

Definition at line 460 of file [virtio-block](#).

## 14.327.3 Member Function Documentation

### 14.327.3.1 finalize\_request()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::finalize_request (
 cxx::unique_ptr< Request > req,
 unsigned sz,
 l4_uint8_t status = L4VIRTIO_BLOCK_S_OK) [inline]
```

Releases resources related to a request and notifies the client.

#### Parameters

|               |                                                                 |
|---------------|-----------------------------------------------------------------|
| <i>req</i>    | Pointer to request that has finished.                           |
| <i>sz</i>     | Number of bytes consumed.                                       |
| <i>status</i> | Status of request (see <a href="#">L4virtio_block_status</a> ). |

This function must be called when an asynchronous request finishes, either successfully or with an error. The status byte in the request must have been set prior to calling it.

Definition at line 523 of file [virtio-block](#).

## 14.327.3.2 get\_writeback()

```
template<typename Ds_data>
L4_uint8_t L4virtio::Svr::Block_dev< Ds_data >::get_writeback () [inline], [protected]
```

Get the writeback field from the configuration space.

## Returns

Value of the writeback field.

Definition at line 404 of file [virtio-block](#).

## 14.327.3.3 process\_request()

```
template<typename Ds_data>
virtual bool L4virtio::Svr::Block_dev< Ds_data >::process_request (
 cxx::unique_ptr< Request > && req) [pure virtual]
```

Implements the actual processing of data in the device.

## Parameters

|            |                              |
|------------|------------------------------|
| <i>req</i> | The request to be processed. |
|------------|------------------------------|

## Returns

If false, no further requests will be scheduled.

Synchronous and asynchronous processing of the data is supported. For asynchronous mode, the function should set up the worker and then return false. In synchronous mode, the function should return true, once processing is complete. If there is an error and processing is aborted, the status flag of *req* needs to be set accordingly and the request immediately finished with *finish\_request()* if the client is to be answered.

## 14.327.3.4 register\_obj()

```
template<typename Ds_data>
L4::Cap<void> L4virtio::Svr::Block_dev< Ds_data >::register_obj (
 L4::Registry_iface * registry,
 char const * service = 0) [inline]
```

Attach device to an object registry.

## Parameters

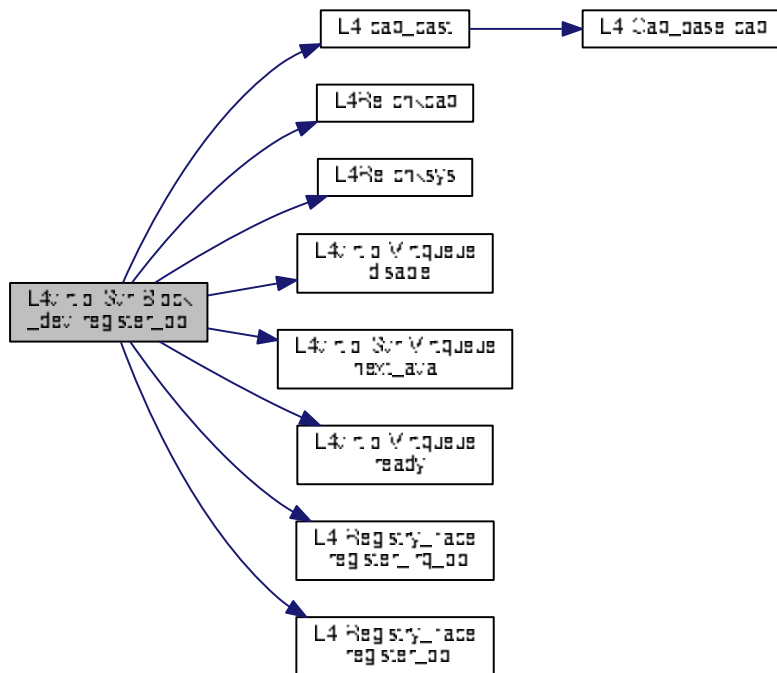
|                 |                                                                    |
|-----------------|--------------------------------------------------------------------|
| <i>registry</i> | Object registry that will be responsible for dispatching requests. |
| <i>service</i>  | Name of an existing capability the device should use.              |

This functions registers the general virtio interface as well as the interrupt handler which is used for receiving client notifications.

Definition at line 557 of file [virtio-block](#).

References [L4::cap\\_cast\(\)](#), [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Virtqueue::disable\(\)](#), [L4\\_EINVAL](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [L4virtio::Virtqueue::ready\(\)](#), [L4::Registry\\_iface::register\\_irq\\_obj\(\)](#), and [L4::Registry\\_iface::register\\_obj\(\)](#).

Here is the call graph for this function:



#### 14.327.3.5 reset\_client()

```
template<typename Ds_data>
virtual bool L4virtio::Svr::Block_dev< Ds_data >::reset_client () [inline], [virtual]
```

The client requests reinitialisation of the connection.

#### Returns

False if reinitialisation is not supported.

True if reinitialisation is supported and the client has been reinitialised successfully.

Definition at line 505 of file [virtio-block](#).

#### 14.327.3.6 set\_blk\_size()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_blk_size (
 14_uint32_t sz) [inline], [protected]
```

Sets block disk size to be reported to the client.

Setting this does not change the logical sector size used for addressing the device.

Definition at line 348 of file [virtio-block](#).

#### 14.327.3.7 set\_config\_wce()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_config_wce (
 14_uint8_t writeback) [inline], [protected]
```

Sets cache mode and enables the the writeback toggle.

##### Parameters

|                  |                                                          |
|------------------|----------------------------------------------------------|
| <i>writeback</i> | Mode of the cache (0 for writethrough, 1 for writeback). |
|------------------|----------------------------------------------------------|

Definition at line 391 of file [virtio-block](#).

#### 14.327.3.8 set\_discard()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_discard (
 14_uint32_t max_discard_sectors,
 14_uint32_t max_discard_seg,
 14_uint32_t discard_sector_alignment) [inline], [protected]
```

Sets constraints for and enables the discard command.

##### Parameters

|                                 |                                                                        |
|---------------------------------|------------------------------------------------------------------------|
| <i>max_discard_sectors</i>      | Maximum discard sectors size.                                          |
| <i>max_discard_seg</i>          | Maximum discard segment number.                                        |
| <i>discard_sector_alignment</i> | Can be used by the driver when splitting a request based on alignment. |

Definition at line 418 of file [virtio-block](#).

**14.327.3.9 set\_flush()**

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_flush () [inline], [protected]
```

Enables the flush command.

Definition at line 380 of file [virtio-block](#).

**14.327.3.10 set\_size\_max()**

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_size_max (
 l4_uint32_t sz) [inline], [protected]
```

Sets the maximum size of any single segment reported to client.

The limit is also applied to any incoming requests. Requests with larger segments result in an IO error being reported to the client. That means that [process\\_request\(\)](#) can safely make the assumption that all segments in the received request are smaller.

Definition at line 306 of file [virtio-block](#).

**14.327.3.11 set\_topology()**

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_topology (
 l4_uint8_t physical_block_exp,
 l4_uint8_t alignment_offset,
 l4_uint32_t min_io_size,
 l4_uint32_t opt_io_size) [inline], [protected]
```

Sets the I/O alignment information reported back to the client.

**Parameters**

|                           |                                                   |
|---------------------------|---------------------------------------------------|
| <i>physical_block_exp</i> | Number of logical blocks per physical block(log2) |
| <i>alignment_offset</i>   | Offset of the first aligned logical block         |
| <i>min_io_size</i>        | Suggested minimum I/O size in blocks              |
| <i>opt_io_size</i>        | Optimal I/O size in blocks                        |

Definition at line 364 of file [virtio-block](#).

## 14.327.3.12 set\_write\_zeroes()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_write_zeroes (
 l4_uint32_t max_write_zeroes_sectors,
 l4_uint32_t max_write_zeroes_seg,
 l4_uint8_t write_zeroes_may_unmap) [inline], [protected]
```

Sets constraints for and enables the write zeroes command.

## Parameters

|                                 |                                                                               |
|---------------------------------|-------------------------------------------------------------------------------|
| <i>max_write_zeroes_sectors</i> | Maximum write zeroes sectors size.                                            |
| <i>max_write_zeroes_seg</i>     | maximum write zeroes segment number.                                          |
| <i>write_zeroes_may_unmap</i>   | Set if a write zeroes request can result in deallocating one or more sectors. |

Definition at line 438 of file [virtio-block](#).

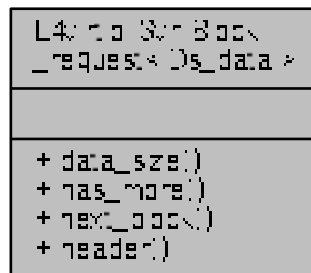
The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

## 14.328 L4virtio::Svr::Block\_request&lt; Ds\_data &gt; Class Template Reference

A request to read or write data.

Collaboration diagram for L4virtio::Svr::Block\_request< Ds\_data >:



## Public Member Functions

- unsigned [data\\_size](#) () const  
*Compute the total size of the data in the request.*
- bool [has\\_more](#) ()  
*Check if the request contains more data blocks.*
- Data\_block [next\\_block](#) ()  
*Return next block in scatter-gather list.*
- [l4virtio\\_block\\_header\\_t](#) const & [header](#) () const  
*Return the block request header.*

### 14.328.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_request< Ds_data >
```

A request to read or write data.

Definition at line 29 of file [virtio-block](#).

### 14.328.2 Member Function Documentation

#### 14.328.2.1 data\_size()

```
template<typename Ds_data >
unsigned L4virtio::Svr::Block_request< Ds_data >::data_size () const [inline]
```

Compute the total size of the data in the request.

#### Return values

|             |                                      |
|-------------|--------------------------------------|
| <i>Size</i> | in bytes or 0 if there was an error. |
|-------------|--------------------------------------|

#### Exceptions

|                                            |                           |
|--------------------------------------------|---------------------------|
| <a href="#">L4::Runtime_error(-L4_EIO)</a> | Request has a bad format. |
|--------------------------------------------|---------------------------|

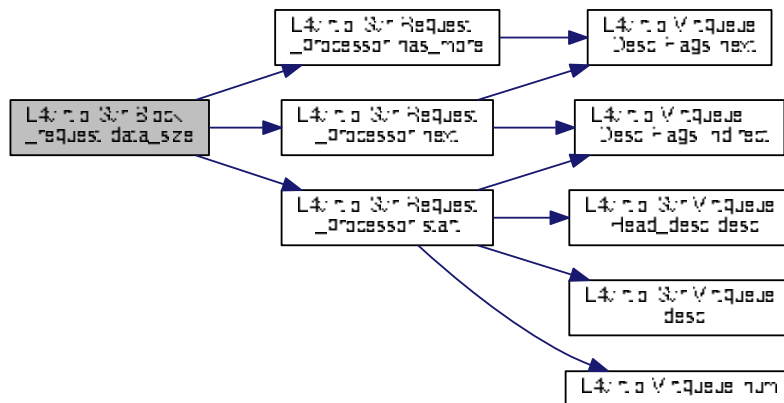
Note that this operation is relatively expensive as it has to iterate over the complete list of blocks.

Definition at line 64 of file [virtio-block](#).

References [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), [L4\\_EIO](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).



Here is the call graph for this function:



#### 14.328.2.2 next\_block()

```
template<typename Ds_data >
Data_block L4virtio::Svr::Block_request< Ds_data >::next_block () [inline]
```

Return next block in scatter-gather list.

#### Returns

Information about the next data block.

#### Exceptions

|                                          |                                  |
|------------------------------------------|----------------------------------|
| <a href="#"><i>L4::Runtime_error</i></a> | No more data block is available. |
| <a href="#"><i>Bad_descriptor</i></a>    | Virtio request is corrupted.     |

Definition at line 114 of file [virtio-block](#).

References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), and [L4\\_EEXIST](#).

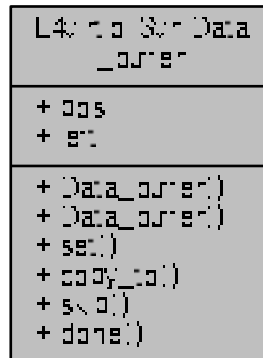
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/virtio-block](#)

## 14.329 L4virtio::Svr::Data\_buffer Struct Reference

Abstract data buffer.

Collaboration diagram for L4virtio::Svr::Data\_buffer:



## Public Member Functions

- `template<typename T >`  
`Data_buffer` (T \*p)  
*Create buffer for object p.*
- `template<typename T >`  
`void set` (T \*p)  
*Set buffer for object p.*
- `l4_uint32_t copy_to` (`Data_buffer` \*dst, `l4_uint32_t` max=UINT\_MAX)  
*Copy contents from this buffer to the destination buffer.*
- `l4_uint32_t skip` (`l4_uint32_t` bytes)  
*Skip given number of bytes in this buffer.*
- `bool done` () const  
*Check if there are no more bytes left in the buffer.*

## Data Fields

- `char * pos`  
*Current buffer position.*
- `l4_uint32_t left`  
*Bytes left in buffer.*

### 14.329.1 Detailed Description

Abstract data buffer.

Definition at line 246 of file `virtio`.

## 14.329.2 Constructor & Destructor Documentation

### 14.329.2.1 Data\_buffer()

```
template<typename T >
L4virtio::Svr::Data_buffer::Data_buffer (
 T * p) [inline], [explicit]
```

Create buffer for object *p*.

#### Template Parameters

|          |                           |
|----------|---------------------------|
| <i>T</i> | type of object (implicit) |
|----------|---------------------------|

#### Parameters

|          |                    |
|----------|--------------------|
| <i>p</i> | pointer to object. |
|----------|--------------------|

The buffer shall point to the start of the object *p* and the size left is sizeof(T).

Definition at line 262 of file [virtio](#).

## 14.329.3 Member Function Documentation

### 14.329.3.1 copy\_to()

```
l4_uint32_t L4virtio::Svr::Data_buffer::copy_to (
 Data_buffer * dst,
 l4_uint32_t max = UINT_MAX) [inline]
```

Copy contents from this buffer to the destination buffer.

#### Parameters

|            |                                             |
|------------|---------------------------------------------|
| <i>dst</i> | Destination buffer.                         |
| <i>max</i> | (optional) Maximum number of bytes to copy. |

#### Returns

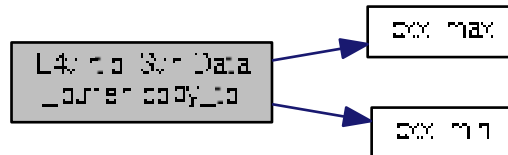
the number of bytes copied.

This function copies at most *max* bytes from this to *dst*. If *max* is omitted, copies the maximum number of bytes available that fit *dst*.

Definition at line 291 of file [virtio](#).

References [left](#), [cxx::max\(\)](#), [cxx::min\(\)](#), and [pos](#).

Here is the call graph for this function:



#### 14.329.3.2 done()

```
bool L4virtio::Svr::Data_buffer::done () const [inline]
```

Check if there are no more bytes left in the buffer.

##### Returns

true if there are no more bytes left in the buffer.

Definition at line 323 of file [virtio](#).

#### 14.329.3.3 set()

```
template<typename T >
void L4virtio::Svr::Data_buffer::set (
 T * p) [inline]
```

Set buffer for object *p*.

##### Template Parameters

|          |                           |
|----------|---------------------------|
| <i>T</i> | type of object (implicit) |
|----------|---------------------------|

##### Parameters

|          |                    |
|----------|--------------------|
| <i>p</i> | pointer to object. |
|----------|--------------------|

The buffer shall point to the start of the object *p* and the size left is sizeof(T).

Definition at line 275 of file [virtio](#).

#### 14.329.3.4 skip()

```
l4_uint32_t L4virtio::Svr::Data_buffer::skip (
 l4_uint32_t bytes) [inline]
```

Skip given number of bytes in this buffer.

##### Parameters

|              |                                        |
|--------------|----------------------------------------|
| <i>bytes</i> | Number of bytes that shall be skipped. |
|--------------|----------------------------------------|

##### Returns

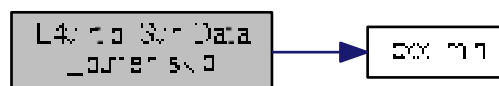
The number of bytes skipped.

Try to skip the given number of bytes in this buffer, if there are less bytes left in the buffer that given then at most left bytes are skipped and the amount is returned.

Definition at line 311 of file [virtio](#).

References [cxx::min\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

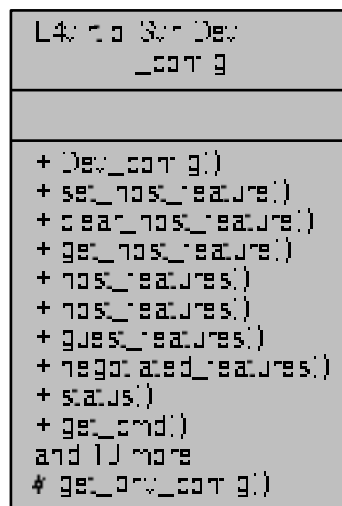
- `I4/I4virtio/server/virtio`

### 14.330 L4virtio::Svr::Dev\_config Class Reference

Abstraction for L4-Virtio device config memory.

Inherited by L4virtio::Svr::Dev\_config\_t< l4virtio\_block\_config\_t >, and L4virtio::Svr::Dev\_config\_t< PRIV\_CONFIG >.

Collaboration diagram for L4virtio::Svr::Dev\_config:



#### Public Member Functions

- [Dev\\_config](#) ([l4\\_uint32\\_t](#) vendor, [l4\\_uint32\\_t](#) device, unsigned cfg\_size, [l4\\_uint32\\_t](#) num\_queues=0)  
*Setup/Create a L4-Virtio config data space.*
- [l4\\_uint32\\_t](#) [guest\\_features](#) (unsigned idx) const  
*Return a specific set of guest features.*
- [l4\\_uint32\\_t](#) [negotiated\\_features](#) (unsigned idx) const  
*Compute a specific set of negotiated features.*
- [Status](#) [status](#) () const  
*Get current device status (trusted).*
- [l4\\_uint32\\_t](#) [get\\_cmd](#) () const  
*Get the value from the cmd register.*
- void [reset\\_cmd](#) ()  
*Reset the cmd register after execution of a command.*
- void [set\\_status](#) ([Status](#) status)  
*Set device status register.*
- void [set\\_failed](#) ()  
*Set device status failed bit.*
- bool [change\\_queue\\_config](#) ([l4\\_uint32\\_t](#) num\_queues)  
*Setup new queue configuration.*

- [l4virtio\\_config\\_queue\\_t](#) volatile const \* [qconfig](#) (unsigned index) const  
*Get queue read-only config data for queue with the given index.*
- void [reset\\_hdr](#) (bool inc\_generation=false) const  
*Reset the config header to the initial contents.*
- bool [reset\\_queue](#) (unsigned index, unsigned num\_max, bool inc\_generation=false) const  
*Reset queue config for the given queue.*
- [l4virtio\\_config\\_hdr\\_t](#) const volatile \* [hdr](#) () const  
*Get a read-only pointer to the config header.*
- [L4::Cap](#)< [L4Re::Dataspace](#) > [ds](#) () const  
*Get data-space capability for the shared config data space.*
- [l4\\_addr\\_t](#) [ds\\_offset](#) () const  
*Return the offset into the config dataspace where the device configuration starts.*

### 14.330.1 Detailed Description

Abstraction for L4-Virtio device config memory.

Virtio defines a device configuration mechanism, L4-Virtio implements this mechanism based on shared memory a [set\\_status\(\)](#) and a [config\\_queue\(\)](#) call. This class provides an abstraction for L4-Virtio host implementations to establish such a shared memory data space and providing the necessary contents and access functions.

Definition at line 56 of file [l4virtio](#).

### 14.330.2 Constructor & Destructor Documentation

#### 14.330.2.1 Dev\_config()

```
L4virtio::Svr::Dev_config::Dev_config (
 l4_uint32_t vendor,
 l4_uint32_t device,
 unsigned cfg_size,
 l4_uint32_t num_queues = 0) [inline]
```

Setup/Create a L4-Virtio config data space.

#### Parameters

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>vendor</i>     | The vendor ID to store in config header.              |
| <i>device</i>     | The device ID to store in config header.              |
| <i>cfg_size</i>   | The size of the device-specific config data in bytes. |
| <i>num_queues</i> | The number of queues provided by the device.          |

This constructor allocates a data space used for L4-virtio config attaches the data space to the local address space and writes the initial contents to the config header.

Definition at line 97 of file [l4virtio](#).

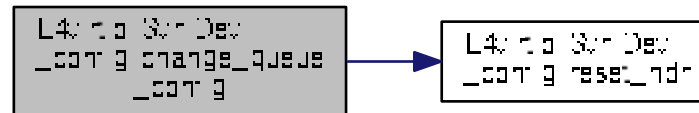




Definition at line 234 of file [l4virtio](#).

References [L4\\_PAGESIZE](#), and [reset\\_hdr\(\)](#).

Here is the call graph for this function:



#### 14.330.3.2 ds()

[L4::Cap<L4Re::Dataspace>](#) L4virtio::Svr::Dev\_config::ds ( ) const [inline]

Get data-space capability for the shared config data space.

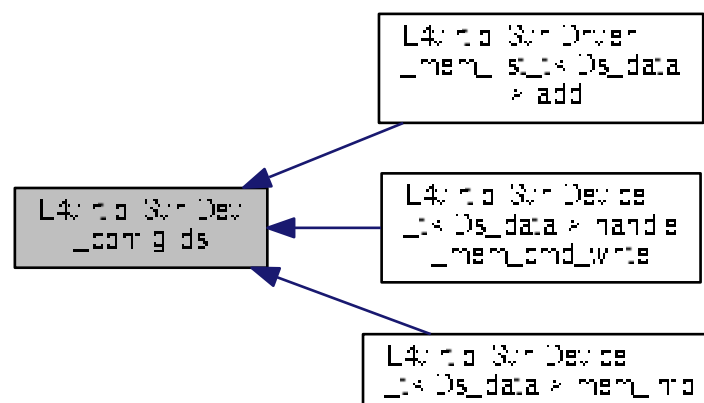
##### Returns

Capability for the shared config data space.

Definition at line 323 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t<Ds\\_data>::add\(\)](#), [L4virtio::Svr::Device\\_t<Ds\\_data>::handle\\_mem\\_cmd\\_write\(\)](#), and [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#).

Here is the caller graph for this function:



### 14.330.3.3 get\_cmd()

```
14_uint32_t L4virtio::Svr::Dev_config::get_cmd () const [inline]
```

Get the value from the `cmd` register.

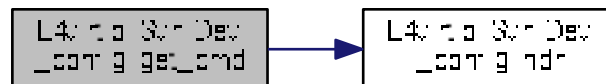
Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 189 of file [l4virtio](#).

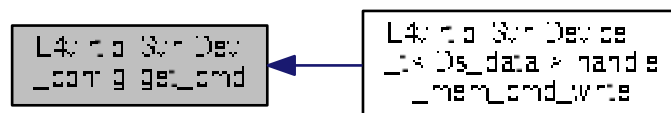
References [l4virtio\\_config\\_hdr\\_t::cmd](#), and [hdr\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::handle\\_mem\\_cmd\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.330.3.4 guest\_features()

```
14_uint32_t L4virtio::Svr::Dev_config::guest_features (
 unsigned idx) const [inline]
```

Return a specific set of guest features.

## Parameters

|            |                                      |
|------------|--------------------------------------|
| <i>idx</i> | Index into the guest features array. |
|------------|--------------------------------------|

## Return values

|            |                                 |
|------------|---------------------------------|
| <i>The</i> | selected set of guest features. |
|------------|---------------------------------|

This function returns a specific 32bit set of features enabled by the guest/driver. *idx* is the index in the guest features array, resp. the 32 bit set to return.

Definition at line 157 of file [l4virtio](#).

14.330.3.5 `hdr()`

```
l4virtio_config_hdr_t const volatile* L4virtio::Svr::Dev_config::hdr () const [inline]
```

Get a read-only pointer to the config header.

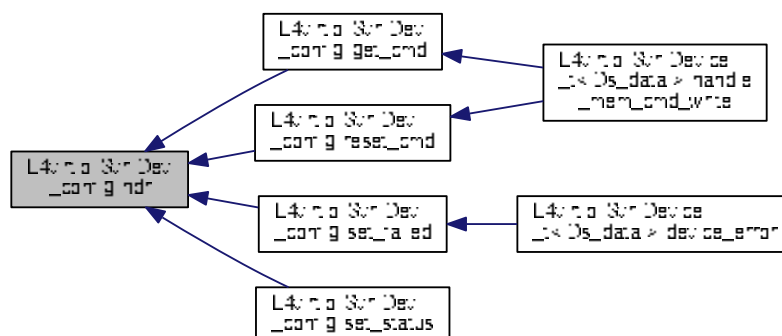
## Returns

Read-only pointer to the shared config header.

Definition at line 316 of file [l4virtio](#).

Referenced by [get\\_cmd\(\)](#), [reset\\_cmd\(\)](#), [set\\_failed\(\)](#), and [set\\_status\(\)](#).

Here is the caller graph for this function:

14.330.3.6 `negotiated_features()`

```
l4_uint32_t L4virtio::Svr::Dev_config::negotiated_features (
 unsigned idx) const [inline]
```

Compute a specific set of negotiated features.

## Parameters

|            |                                           |
|------------|-------------------------------------------|
| <i>idx</i> | Index into the guest/host features array. |
|------------|-------------------------------------------|

## Return values

|            |                                      |
|------------|--------------------------------------|
| <i>The</i> | selected set of negotiated features. |
|------------|--------------------------------------|

This function returns a specific 32-bit set of features negotiated by the guest/driver and host/device. *idx* is the index in the guest/host features array, resp. the 32-bit set to return.

Definition at line 171 of file [l4virtio](#).

14.330.3.7 `qconfig()`

```
l4virtio_config_queue_t volatile const* L4virtio::Svr::Dev_config::qconfig (
 unsigned index) const [inline]
```

Get queue read-only config data for queue with the given *index*.

## Parameters

|              |                         |
|--------------|-------------------------|
| <i>index</i> | The index of the queue. |
|--------------|-------------------------|

## Returns

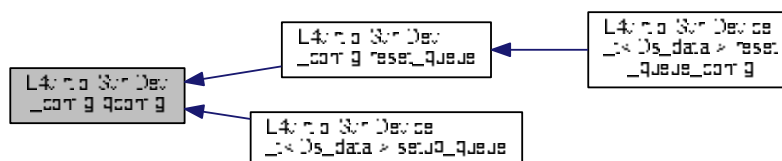
Read-only pointer to the config of the queue with the given *index*, or NULL if *index* is out of range.

Definition at line 251 of file [l4virtio](#).

References [L4\\_UNLIKELY](#).

Referenced by [reset\\_queue\(\)](#), and [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.330.3.8 reset\_cmd()

```
void L4virtio::Svr::Dev_config::reset_cmd () [inline]
```

Reset the `cmd` register after execution of a command.

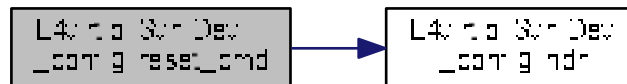
This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 200 of file [l4virtio](#).

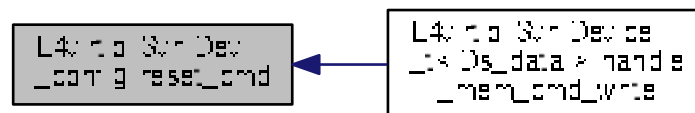
References [l4virtio\\_config\\_hdr\\_t::cmd](#), and [hdr\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::handle\\_mem\\_cmd\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 14.330.3.9 reset\_queue()

```
bool L4virtio::Svr::Dev_config::reset_queue (
 unsigned index,
 unsigned num_max,
 bool inc_generation = false) const [inline]
```

Reset queue config for the given queue.

## Parameters

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <i>index</i>          | The index of the queue to reset.                             |
| <i>num_max</i>        | The maximum number of descriptor supported by this queue.    |
| <i>inc_generation</i> | The config generation will be incremented when this is true. |

## Returns

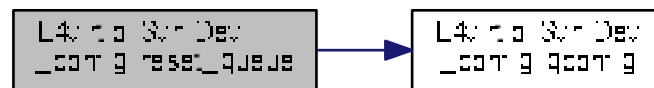
true on success, or false when *index* is out of range.

Definition at line 293 of file [l4virtio](#).

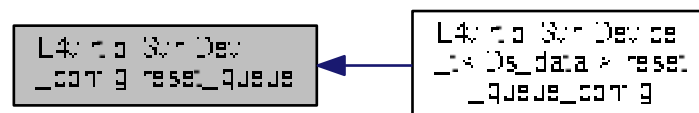
References [L4\\_UNLIKELY](#), [l4virtio\\_config\\_queue\\_t::num](#), [l4virtio\\_config\\_queue\\_t::num\\_max](#), [qconfig\(\)](#), and [l4virtio\\_config\\_queue\\_t::ready](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::reset\\_queue\\_config\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 14.330.3.10 set\_failed()

```
void L4virtio::Svr::Dev_config::set_failed () [inline]
```

Set device status failed bit.

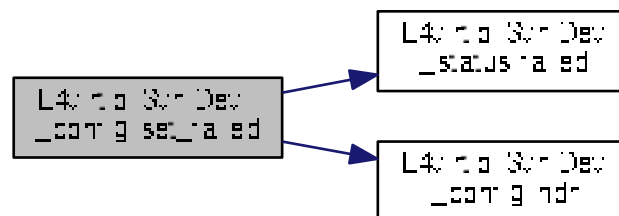
This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 224 of file [l4virtio](#).

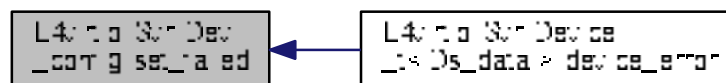
References [L4virtio::Svr::Dev\\_status::failed\(\)](#), [hdr\(\)](#), [L4virtio::Svr::Dev\\_status::raw](#), and [l4virtio\\_config\\_hdr\\_t::status](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::device\\_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 14.330.3.11 set\_status()

```
void L4virtio::Svr::Dev_config::set_status (
 Status status) [inline]
```

Set device status register.

## Parameters

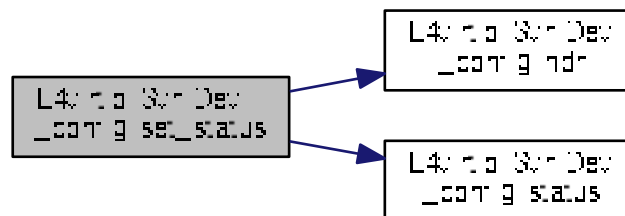
|               |                                               |
|---------------|-----------------------------------------------|
| <i>status</i> | The new value for the device status register. |
|---------------|-----------------------------------------------|

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 212 of file [l4virtio](#).

References [hdr\(\)](#), [L4virtio::Svr::Dev\\_status::raw](#), [l4virtio\\_config\\_hdr\\_t::status](#), and [status\(\)](#).

Here is the call graph for this function:



## 14.330.3.12 status()

```
Status L4virtio::Svr::Dev_config::status () const [inline]
```

Get current device status (trusted).

## Returns

Current device status register (trusted).

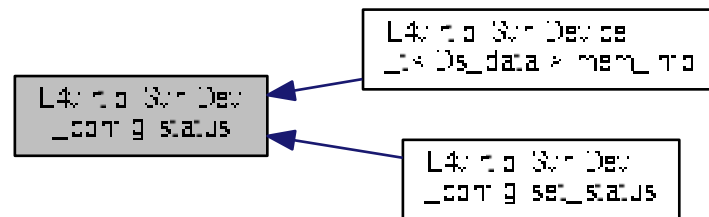
The status returned by this function is value stored internally and cannot be written by the guest (i.e., the value can be taken as trusted.)

Definition at line 181 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#), and [set\\_status\(\)](#).



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

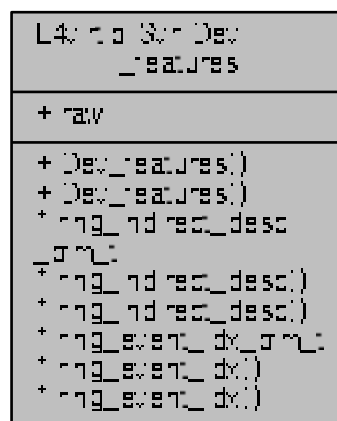
- `I4/I4virtio/server/I4virtio`

## 14.331 L4virtio::Svr::Dev\_features Struct Reference

Type for device feature bitmap.

Inherited by `L4virtio::Svr::Block_features`.

Collaboration diagram for `L4virtio::Svr::Dev_features`:



### Public Member Functions

- [Dev\\_features\(I4\\_uint32\\_t v\)](#)  
*Make Features from a raw bitmap.*

## Data Fields

- [l4\\_uint32\\_t raw](#)  
*The raw value of the features bitmap.*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 28, 28 > [ring\\_indirect\\_desc\\_bfm\\_t](#)  
*Type to access the `ring_indirect_desc` bits ( 28 to 28 ) of `raw` .*
- [ring\\_indirect\\_desc\\_bfm\\_t::Val ring\\_indirect\\_desc \(\)](#) const  
*Get the `ring_indirect_desc` bits ( 28 to 28 ) of `raw` .*
- [ring\\_indirect\\_desc\\_bfm\\_t::Ref ring\\_indirect\\_desc \(\)](#)  
*Get a reference to the `ring_indirect_desc` bits ( 28 to 28 ) of `raw` .*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 29, 29 > [ring\\_event\\_idx\\_bfm\\_t](#)  
*Type to access the `ring_event_idx` bits ( 29 to 29 ) of `raw` .*
- [ring\\_event\\_idx\\_bfm\\_t::Val ring\\_event\\_idx \(\)](#) const  
*Get the `ring_event_idx` bits ( 29 to 29 ) of `raw` .*
- [ring\\_event\\_idx\\_bfm\\_t::Ref ring\\_event\\_idx \(\)](#)  
*Get a reference to the `ring_event_idx` bits ( 29 to 29 ) of `raw` .*

### 14.331.1 Detailed Description

Type for device feature bitmap.

Definition at line 75 of file [virtio](#).

### 14.331.2 Member Typedef Documentation

#### 14.331.2.1 ring\_event\_idx\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 29 , 29 > L4virtio::Svr::Dev_features::ring_event_idx↵
_bfm_t
```

Type to access the `ring_event_idx` bits ( 29 to 29 ) of `raw` .

Definition at line 83 of file [virtio](#).

#### 14.331.2.2 ring\_indirect\_desc\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 28 , 28 > L4virtio::Svr::Dev_features::ring_indirect_↵
desc_bfm_t
```

Type to access the `ring_indirect_desc` bits ( 28 to 28 ) of `raw` .

Definition at line 83 of file [virtio](#).

### 14.331.3 Member Function Documentation

#### 14.331.3.1 ring\_event\_idx() [1/2]

```
ring_event_idx_bfm_t::Val L4virtio::Svr::Dev_features::ring_event_idx () const [inline]
```

Get the *ring\_event\_idx* bits ( 29 to 29 ) of *raw* .

Definition at line 84 of file [virtio](#).

#### 14.331.3.2 ring\_event\_idx() [2/2]

```
ring_event_idx_bfm_t::Ref L4virtio::Svr::Dev_features::ring_event_idx () [inline]
```

Get a reference to the *ring\_event\_idx* bits ( 29 to 29 ) of *raw* .

Definition at line 84 of file [virtio](#).

#### 14.331.3.3 ring\_indirect\_desc() [1/2]

```
ring_indirect_desc_bfm_t::Val L4virtio::Svr::Dev_features::ring_indirect_desc () const [inline]
```

Get the *ring\_indirect\_desc* bits ( 28 to 28 ) of *raw* .

Definition at line 83 of file [virtio](#).

#### 14.331.3.4 ring\_indirect\_desc() [2/2]

```
ring_indirect_desc_bfm_t::Ref L4virtio::Svr::Dev_features::ring_indirect_desc () [inline]
```

Get a reference to the *ring\_indirect\_desc* bits ( 28 to 28 ) of *raw* .

Definition at line 83 of file [virtio](#).

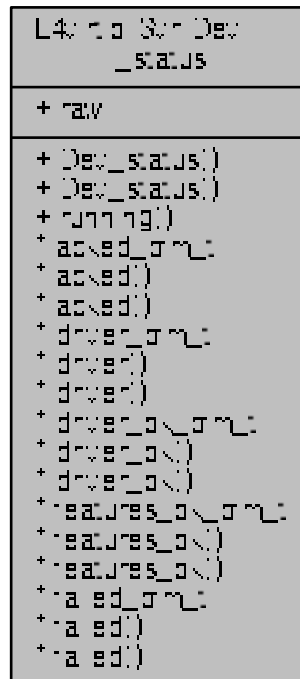
The documentation for this struct was generated from the following file:

- [l4/l4virtio/server/virtio](#)

## 14.332 L4virtio::Svr::Dev\_status Struct Reference

Type of the device status register.

Collaboration diagram for L4virtio::Svr::Dev\_status:



### Public Member Functions

- [Dev\\_status](#) ([l4\\_uint32\\_t](#) v)  
*Make Status from raw value.*
- bool [running](#) () const  
*Check if the device is in running state.*

### Data Fields

- unsigned char [raw](#)  
*Raw value of the VIRTIO device status register.*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > [acked\\_bfm\\_t](#)  
*Type to access the acked bits ( 0 to 0 ) of raw .*
- [acked\\_bfm\\_t::Val](#) [acked](#) () const  
*Get the acked bits ( 0 to 0 ) of raw .*
- [acked\\_bfm\\_t::Ref](#) [acked](#) ()

Get a reference to the *acked* bits ( 0 to 0 ) of *raw* .

- typedef [cxx::Bitfield](#)< decltype([raw](#)), 1, 1 > [driver\\_bfm\\_t](#)  
Type to access the *driver* bits ( 1 to 1 ) of *raw* .
- [driver\\_bfm\\_t::Val driver](#) () const  
Get the *driver* bits ( 1 to 1 ) of *raw* .
- [driver\\_bfm\\_t::Ref driver](#) ()  
Get a reference to the *driver* bits ( 1 to 1 ) of *raw* .
  
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 2, 2 > [driver\\_ok\\_bfm\\_t](#)  
Type to access the *driver\_ok* bits ( 2 to 2 ) of *raw* .
- [driver\\_ok\\_bfm\\_t::Val driver\\_ok](#) () const  
Get the *driver\_ok* bits ( 2 to 2 ) of *raw* .
- [driver\\_ok\\_bfm\\_t::Ref driver\\_ok](#) ()  
Get a reference to the *driver\_ok* bits ( 2 to 2 ) of *raw* .
  
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 3, 3 > [features\\_ok\\_bfm\\_t](#)  
Type to access the *features\_ok* bits ( 3 to 3 ) of *raw* .
- [features\\_ok\\_bfm\\_t::Val features\\_ok](#) () const  
Get the *features\_ok* bits ( 3 to 3 ) of *raw* .
- [features\\_ok\\_bfm\\_t::Ref features\\_ok](#) ()  
Get a reference to the *features\_ok* bits ( 3 to 3 ) of *raw* .
  
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 7, 7 > [failed\\_bfm\\_t](#)  
Type to access the *failed* bits ( 7 to 7 ) of *raw* .
- [failed\\_bfm\\_t::Val failed](#) () const  
Get the *failed* bits ( 7 to 7 ) of *raw* .
- [failed\\_bfm\\_t::Ref failed](#) ()  
Get a reference to the *failed* bits ( 7 to 7 ) of *raw* .

### 14.332.1 Detailed Description

Type of the device status register.

Definition at line 44 of file [virtio](#).

### 14.332.2 Member Typedef Documentation

#### 14.332.2.1 [acked\\_bfm\\_t](#)

```
typedef cxx::Bitfield<decltype(raw), 0 , 0 > L4virtio::Svr::Dev_status::acked_bfm_t
```

Type to access the *acked* bits ( 0 to 0 ) of *raw* .

Definition at line 52 of file [virtio](#).

#### 14.332.2.2 driver\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 1 , 1 > L4virtio::Svr::Dev_status::driver_bfm_t
```

Type to access the *driver* bits ( 1 to 1 ) of *raw* .

Definition at line 52 of file [virtio](#).

#### 14.332.2.3 driver\_ok\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 2 , 2 > L4virtio::Svr::Dev_status::driver_ok_bfm_t
```

Type to access the *driver\_ok* bits ( 2 to 2 ) of *raw* .

Definition at line 53 of file [virtio](#).

#### 14.332.2.4 failed\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 7 , 7 > L4virtio::Svr::Dev_status::failed_bfm_t
```

Type to access the *failed* bits ( 7 to 7 ) of *raw* .

Definition at line 55 of file [virtio](#).

#### 14.332.2.5 features\_ok\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 3 , 3 > L4virtio::Svr::Dev_status::features_ok_bfm_t
```

Type to access the *features\_ok* bits ( 3 to 3 ) of *raw* .

Definition at line 54 of file [virtio](#).

### 14.332.3 Member Function Documentation

14.332.3.1 `acked()` [1/2]

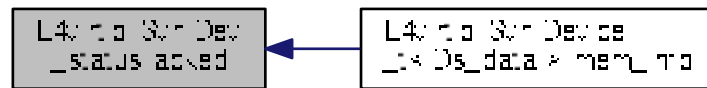
```
acked_bfm_t::Val L4virtio::Svr::Dev_status::acked () const [inline]
```

Get the *acked* bits ( 0 to 0 ) of *raw* .

Definition at line 52 of file [virtio](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#).

Here is the caller graph for this function:

14.332.3.2 `acked()` [2/2]

```
acked_bfm_t::Ref L4virtio::Svr::Dev_status::acked () [inline]
```

Get a reference to the *acked* bits ( 0 to 0 ) of *raw* .

Definition at line 52 of file [virtio](#).

14.332.3.3 `driver()` [1/2]

```
driver_bfm_t::Ref L4virtio::Svr::Dev_status::driver () [inline]
```

Get a reference to the *driver* bits ( 1 to 1 ) of *raw* .

Definition at line 53 of file [virtio](#).

14.332.3.4 `driver()` [2/2]

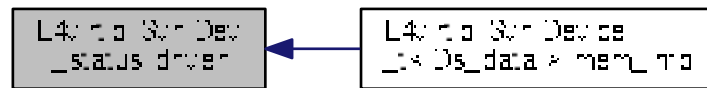
```
driver_bfm_t::Val L4virtio::Svr::Dev_status::driver () const [inline]
```

Get the *driver* bits ( 1 to 1 ) of *raw* .

Definition at line 53 of file [virtio](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#).

Here is the caller graph for this function:

14.332.3.5 `driver_ok()` [1/2]

```
driver_ok_bfm_t::Val L4virtio::Svr::Dev_status::driver_ok () const [inline]
```

Get the *driver\_ok* bits ( 2 to 2 ) of *raw* .

Definition at line 54 of file [virtio](#).

14.332.3.6 `driver_ok()` [2/2]

```
driver_ok_bfm_t::Ref L4virtio::Svr::Dev_status::driver_ok () [inline]
```

Get a reference to the *driver\_ok* bits ( 2 to 2 ) of *raw* .

Definition at line 54 of file [virtio](#).



## 14.332.3.7 failed() [1/2]

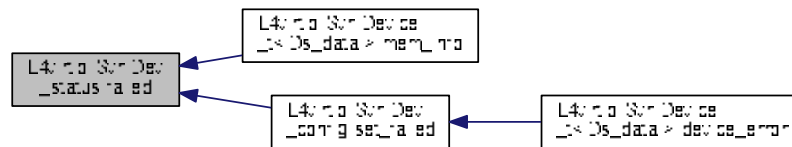
```
failed_bfm_t::Val L4virtio::Svr::Dev_status::failed () const [inline]
```

Get the *failed* bits ( 7 to 7 ) of *raw* .

Definition at line 56 of file [virtio](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#), and [L4virtio::Svr::Dev\\_config::set\\_failed\(\)](#).

Here is the caller graph for this function:



## 14.332.3.8 failed() [2/2]

```
failed_bfm_t::Ref L4virtio::Svr::Dev_status::failed () [inline]
```

Get a reference to the *failed* bits ( 7 to 7 ) of *raw* .

Definition at line 56 of file [virtio](#).

## 14.332.3.9 features\_ok() [1/2]

```
features_ok_bfm_t::Ref L4virtio::Svr::Dev_status::features_ok () [inline]
```

Get a reference to the *features\_ok* bits ( 3 to 3 ) of *raw* .

Definition at line 55 of file [virtio](#).

## 14.332.3.10 features\_ok() [2/2]

```
features_ok_bfm_t::Val L4virtio::Svr::Dev_status::features_ok () const [inline]
```

Get the *features\_ok* bits ( 3 to 3 ) of *raw* .

Definition at line 55 of file [virtio](#).

#### 14.332.3.11 `running()`

```
bool L4virtio::Svr::Dev_status::running () const [inline]
```

Check if the device is in running state.

##### Returns

true if the device is in running state.

The device is in running state when [acked\(\)](#), [driver\(\)](#), [features\\_ok\(\)](#), and [driver\\_ok\(\)](#) return true, and [failed\(\)](#) returns false.

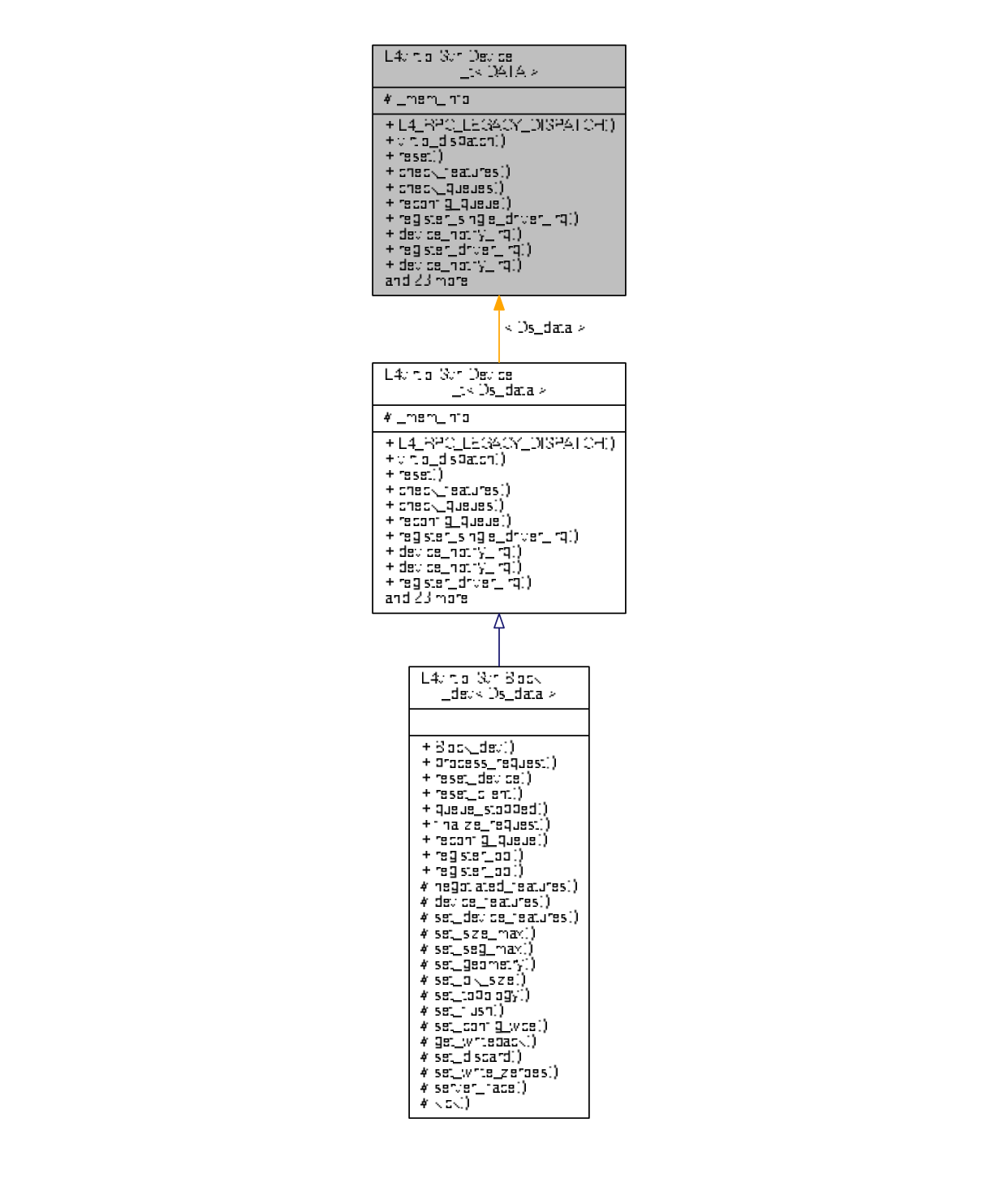
Definition at line [66](#) of file [virtio](#).

The documentation for this struct was generated from the following file:

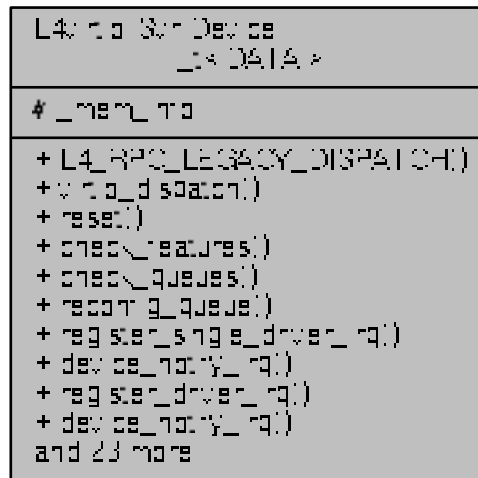
- [l4/l4virtio/server/virtio](#)

## 14.333 `L4virtio::Svr::Device_t< DATA >` Class Template Reference

Server-side L4-VIRTIO device stub.



Collaboration diagram for L4virtio::Svr::Device\_t< DATA >:



## Public Member Functions

- virtual void [reset](#) ()=0  
*reset callback, called for doing a device reset*
- virtual bool [check\\_features](#) ()  
*callback for checking the subset of accepted features*
- virtual bool [check\\_queues](#) ()=0  
*callback for checking if the queues at DRIVER\_OK transition*
- virtual int [reconfig\\_queue](#) (unsigned idx)=0  
*callback for client queue-config request*
- virtual void [register\\_single\\_driver\\_irq](#) ()  
*callback for registering a single guest IRQ for all queues (old-style)*
- virtual L4::Cap< L4::Irq > [device\\_notify\\_irq](#) () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void [register\\_driver\\_irq](#) (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::Irq > [device\\_notify\\_irq](#) (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned [num\\_events\\_supported](#) () const  
*Return the highest notification index supported.*
- [Device\\_t](#) (Dev\_config \*dev\_config)  
*Make a device for the given config.*
- [Mem\\_list](#) const \* [mem\\_info](#) () const  
*Get the memory region list used for this device.*
- void [reset\\_queue\\_config](#) (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void [init\\_mem\\_info](#) (unsigned num)

*Initialize the memory region list to the given maximum.*

- void [device\\_error](#) ()

*Transition device into failed state.*

- bool [setup\\_queue](#) ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)

*Enable/disable the specified queue.*

- bool [handle\\_mem\\_cmd\\_write](#) ()

*Check for a value in the `cmd` register and handle a write.*

## Protected Attributes

- [Mem\\_list\\_mem\\_info](#)

*Memory region list.*

### 14.333.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Device_t< DATA >
```

Server-side L4-VIRTIO device stub.

This stub supports old-style device registration with single IRQs (via [register\\_iface\(\)](#)) and new-style multi-event registration (using [get\\_device\\_config\(\)](#), [bind\(\)](#) and [get\\_device\\_notification\\_irq\(\)](#)).

In their default implementation the callbacks provide a wrapper from old-style to new-style functions, so that legacy devices provide both interfaces without any changes to their implementation.

New devices should always implement the new-style interface. If required, they can also provide a backward-compatibility mode by implementing the old-style interface as well.

The old-style interface is considered deprecated and will be removed at some point.

Definition at line [731](#) of file [l4virtio](#).

### 14.333.2 Member Function Documentation

#### 14.333.2.1 [device\\_error\(\)](#)

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::device_error () [inline]
```

Transition device into failed state.

#### Note

Callers should trigger a guest config IRQ after calling this function.

This function does a full reset, (calls [reset\(\)](#)) and sets the failed bit in the device status register.

Definition at line [966](#) of file [l4virtio](#).

#### 14.333.2.2 device\_notify\_irq()

```
template<typename DATA>
virtual L4::Cap<L4::Irq> L4virtio::Svr::Device_t< DATA >::device_notify_irq (
 unsigned idx) [inline], [virtual]
```

Callback to gather the device notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 791 of file [l4virtio](#).

#### 14.333.2.3 handle\_mem\_cmd\_write()

```
template<typename DATA>
bool L4virtio::Svr::Device_t< DATA >::handle_mem_cmd_write () [inline]
```

Check for a value in the `cmd` register and handle a write.

This function checks for a value in the `cmd` register and executes the command if there is any, or returns false if there was no command.

Execution of the command is signaled by a zero in the `cmd` register.

Definition at line 1086 of file [l4virtio](#).

#### 14.333.2.4 init\_mem\_info()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::init_mem_info (
 unsigned num) [inline]
```

Initialize the memory region list to the given maximum.

##### Parameters

|            |                                                       |
|------------|-------------------------------------------------------|
| <i>num</i> | Maximum number of memory regions that can be managed. |
|------------|-------------------------------------------------------|

Definition at line 953 of file [l4virtio](#).

#### 14.333.2.5 register\_driver\_irq()

```
template<typename DATA>
virtual void L4virtio::Svr::Device_t< DATA >::register_driver_irq (
 unsigned idx) [inline], [virtual]
```

Callback for registering an notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 777 of file [l4virtio](#).

#### 14.333.2.6 reset\_queue\_config()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::reset_queue_config (
 unsigned idx,
 unsigned num_max,
 bool inc_generation = false) [inline]
```

Trigger reset for the configuration space for queue *idx*.

##### Parameters

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| <i>idx</i>            | The queue index to reset.                                    |
| <i>num_max</i>        | Maximum number of entries in this queue.                     |
| <i>inc_generation</i> | The config generation will be incremented when this is true. |

This function resets the driver-readable configuration space for the queue with the given index. The queue configuration is reset to all 0, name *num\_max* to the given value.

Definition at line 943 of file [l4virtio](#).

#### 14.333.2.7 setup\_queue()

```
template<typename DATA>
bool L4virtio::Svr::Device_t< DATA >::setup_queue (
 Virtqueue * q,
 unsigned qn,
 unsigned num_max) [inline]
```

Enable/disable the specified queue.

##### Parameters

|                |                                                               |
|----------------|---------------------------------------------------------------|
| <i>q</i>       | Pointer to the ring that represents the virtqueue internally. |
| <i>qn</i>      | Index of the queue.                                           |
| <i>num_max</i> | Maximum number of supported entries in this queue.            |

##### Returns

true for success.

- This function calculates the parameters of the virtqueue from the clients configuration space values, checks the accessibility of the queue data structures and initializes *q* to ready state when all checks succeeded.

Definition at line 985 of file [l4virtio](#).

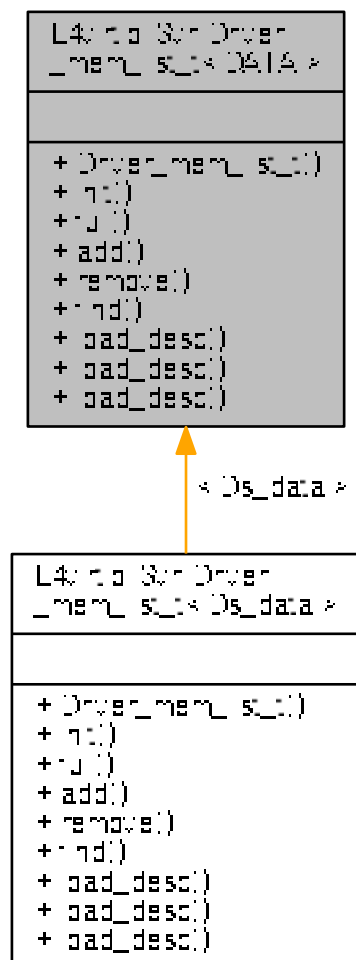
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/l4virtio](#)

### 14.334 L4virtio::Svr::Driver\_mem\_list\_t< DATA > Class Template Reference

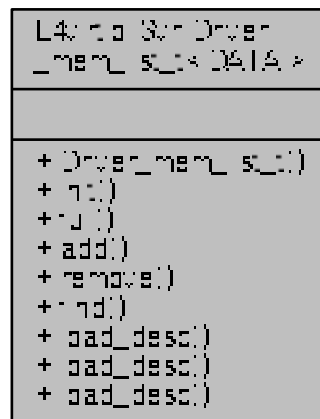
List of driver memory regions assigned to a single L4-VIRTIO transport instance.

Inheritance diagram for L4virtio::Svr::Driver\_mem\_list\_t< DATA >:





Collaboration diagram for L4virtio::Svr::Driver\_mem\_list\_t< DATA >:



## Public Types

- typedef [L4Re::Util::Unique\\_cap< L4Re::Dataspace >](#) [Ds\\_cap](#)  
type for storing a data-space capability internally

## Public Member Functions

- [Driver\\_mem\\_list\\_t](#) ()  
Make an empty, zero capacity list.
- void [init](#) (unsigned max)  
Make a fresh list with capacity max.
- bool [full](#) () const
- [Mem\\_region](#) const \* [add](#) ([l4\\_uint64\\_t](#) drv\_base, [l4\\_umword\\_t](#) size, [l4\\_addr\\_t](#) offset, [Ds\\_cap](#) &&ds)  
Add a new region to the list.
- void [remove](#) ([Mem\\_region](#) const \*r)  
Remove the given region from the list.
- [Mem\\_region](#) \* [find](#) ([l4\\_uint64\\_t](#) base, [l4\\_umword\\_t](#) size) const  
Find memory region containing the given driver address region.
- void [load\\_desc](#) ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, [Virtqueue::Desc](#) const \*\*table) const  
Default implementation for loading an indirect descriptor.
- void [load\\_desc](#) ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, [Mem\\_region](#) const \*\*data) const  
Default implementation returning the Driver\_mem\_region.
- template<typename ARG >  
void [load\\_desc](#) ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, ARG \*data) const  
Default implementation returning generic information.

### 14.334.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Driver_mem_list_t< DATA >
```

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

#### Note

The regions added to this list *must* never overlap.

Definition at line 553 of file [l4virtio](#).

### 14.334.2 Member Function Documentation

#### 14.334.2.1 add()

```
template<typename DATA>
Mem_region const* L4virtio::Svr::Driver_mem_list_t< DATA >::add (
 l4_uint64_t drv_base,
 l4_umword_t size,
 l4_addr_t offset,
 Ds_cap && ds) [inline]
```

Add a new region to the list.

#### Parameters

|                 |                                                            |
|-----------------|------------------------------------------------------------|
| <i>drv_base</i> | Driver base address of the region.                         |
| <i>size</i>     | Size of the region in bytes.                               |
| <i>offset</i>   | Offset within the data space attached to <i>drv_base</i> . |
| <i>ds</i>       | Data space backing the driver memory.                      |

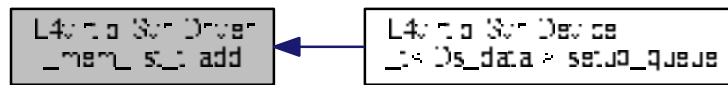
#### Returns

A pointer to the new region.

Definition at line 593 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



#### 14.334.2.2 find()

```

template<typename DATA>
Mem_region* L4virtio::Svr::Driver_mem_list_t< DATA >::find (
 l4_uint64_t base,
 l4_umword_t size) const [inline]

```

Find memory region containing the given driver address region.

##### Parameters

|             |                      |
|-------------|----------------------|
| <i>base</i> | Driver base address. |
| <i>size</i> | Size of the region.  |

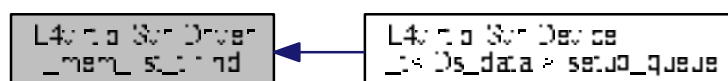
##### Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 627 of file `l4virtio`.

Referenced by `L4virtio::Svr::Device_t< Ds_data >::setup_queue()`.

Here is the caller graph for this function:



## 14.334.2.3 full()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full () const [inline]
```

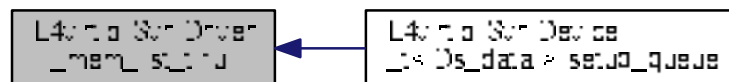
## Returns

True if the remaining capacity is 0.

Definition at line 582 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.334.2.4 init()

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::init (
 unsigned max) [inline]
```

Make a fresh list with capacity *max*.

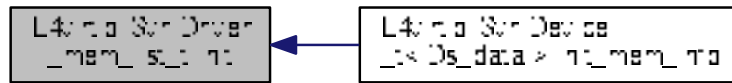
## Parameters

|            |                              |
|------------|------------------------------|
| <i>max</i> | The capacity of this vector. |
|------------|------------------------------|

Definition at line 574 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::init\\_mem\\_info\(\)](#).

Here is the caller graph for this function:



#### 14.334.2.5 load\_desc() [1/3]

```

template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
 Virtqueue::Desc const & desc,
 Request_processor const * p,
 Virtqueue::Desc const ** table) const [inline]

```

Default implementation for loading an indirect descriptor.

##### Parameters

|     |              |                                             |
|-----|--------------|---------------------------------------------|
|     | <i>desc</i>  | The descriptor to load                      |
|     | <i>p</i>     | The request processor calling us            |
| out | <i>table</i> | Shall be set to the loaded descriptor table |

##### Exceptions

|                                       |                                                 |
|---------------------------------------|-------------------------------------------------|
| <a href="#"><i>Bad_descriptor</i></a> | The descriptor address could not be translated. |
|---------------------------------------|-------------------------------------------------|

Definition at line 641 of file [l4virtio](#).

#### 14.334.2.6 load\_desc() [2/3]

```

template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
 Virtqueue::Desc const & desc,
 Request_processor const * p,
 Mem_region const ** data) const [inline]

```

Default implementation returning the Driver\_mem\_region.

## Parameters

|     |             |                                                                                |
|-----|-------------|--------------------------------------------------------------------------------|
|     | <i>desc</i> | The descriptor to load                                                         |
|     | <i>p</i>    | The request processor calling us                                               |
| out | <i>data</i> | Shall be set to a pointer to the Driver_mem_region that covers the descriptor. |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <a href="#">Bad_descriptor</a> | The descriptor address could not be translated. |
|--------------------------------|-------------------------------------------------|

Definition at line 661 of file [l4virtio](#).

## 14.334.2.7 load\_desc() [3/3]

```
template<typename DATA>
template<typename ARG >
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
 Virtqueue::Desc const & desc,
 Request_processor const * p,
 ARG * data) const [inline]
```

Default implementation returning generic information.

## Template Parameters

|            |                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ARG</i> | Abstract argument type used with <a href="#">Request_processor::start()</a> and <a href="#">Request_processor::next()</a> to deliver the result of loading a descriptor. This type must provide a constructor taking three arguments: (1) pointer to a Driver_mem_region, (2) the <a href="#">Virtqueue::Desc</a> descriptor, and (3) a pointer to the calling <a href="#">Request_processor</a> . |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Parameters

|     |             |                                        |
|-----|-------------|----------------------------------------|
|     | <i>desc</i> | The descriptor to load                 |
|     | <i>p</i>    | The request processor calling us       |
| out | <i>data</i> | Shall be assigned to ARG(mem, desc, p) |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <a href="#">Bad_descriptor</a> | The descriptor address could not be translated. |
|--------------------------------|-------------------------------------------------|

Definition at line 688 of file [l4virtio](#).

## 14.334.2.8 remove()

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::remove (
 Mem_region const * r) [inline]
```

Remove the given region from the list.

## Parameters

|          |                                                                                        |
|----------|----------------------------------------------------------------------------------------|
| <i>r</i> | The region to remove (result from <a href="#">add()</a> , or <a href="#">find()</a> ). |
|----------|----------------------------------------------------------------------------------------|

Definition at line 607 of file [l4virtio](#).

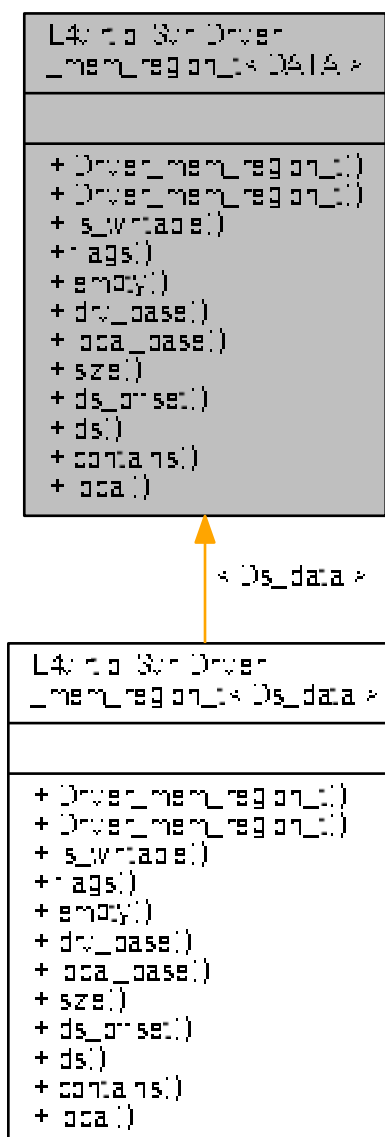
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/l4virtio](#)

## 14.335 L4virtio::Svr::Driver\_mem\_region\_t&lt; DATA &gt; Class Template Reference

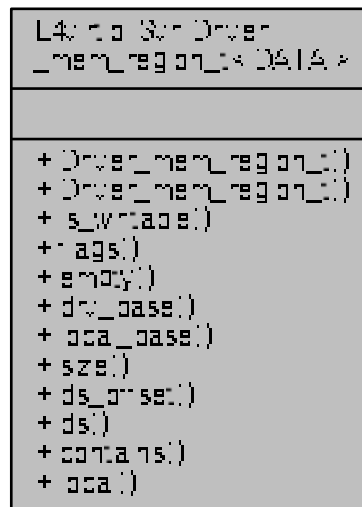
Region of driver memory, that shall be managed locally.

Inheritance diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:





Collaboration diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:



## Public Member Functions

- [Driver\\_mem\\_region\\_t](#) ()  
*Make default empty memroy region.*
- [Driver\\_mem\\_region\\_t](#) (l4\_uint64\_t drv\_base, l4\_umword\_t size, l4\_addr\_t offset, Ds\_cap &&ds)  
*Make a local memory region for the given driver values.*
- bool [is\\_writable](#) () const
- Flags [flags](#) () const
- bool [empty](#) () const
- l4\_uint64\_t [drv\\_base](#) () const
- void \* [local\\_base](#) () const
- l4\_umword\_t [size](#) () const
- l4\_addr\_t [ds\\_offset](#) () const
- L4::Cap< L4Re::Dataspace > [ds](#) () const
- bool [contains](#) (l4\_uint64\_t base, l4\_umword\_t size) const  
*Test if the given driver address range is within this region.*
- template<typename T >  
T \* [local](#) (Ptr< T > p) const  
*Get the local address for driver address p.*

## 14.335.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Driver_mem_region_t< DATA >
```

Region of driver memory, that shall be managed locally.

## Template Parameters

|             |                                       |
|-------------|---------------------------------------|
| <i>DATA</i> | Class defining additional information |
|-------------|---------------------------------------|

Definition at line 380 of file [l4virtio](#).

## 14.335.2 Constructor & Destructor Documentation

### 14.335.2.1 Driver\_mem\_region\_t()

```
template<typename DATA>
L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t (
 l4_uint64_t drv_base,
 l4_umword_t size,
 l4_addr_t offset,
 Ds_cap && ds) [inline]
```

Make a local memory region for the given driver values.

## Parameters

|                 |                                                                                   |
|-----------------|-----------------------------------------------------------------------------------|
| <i>drv_base</i> | Base address of the memory region used by the driver.                             |
| <i>size</i>     | Size of the memory region.                                                        |
| <i>offset</i>   | Offset within the data space that is mapped to <i>drv_base</i> within the driver. |
| <i>ds</i>       | Data space capability backing the memory.                                         |

This constructor attaches the region of given data space to the local address space and stores the corresponding data for later reference.

Definition at line 425 of file [l4virtio](#).

## 14.335.3 Member Function Documentation

### 14.335.3.1 contains()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::contains (
 l4_uint64_t base,
 l4_umword_t size) const [inline]
```

Test if the given driver address range is within this region.

**Parameters**

|             |                                   |
|-------------|-----------------------------------|
| <i>base</i> | The driver base address.          |
| <i>size</i> | The size of the region to lookup. |

**Returns**

true if the given driver address region is contained in this region, false else.

Definition at line 516 of file [l4virtio](#).

**14.335.3.2 drv\_base()**

```
template<typename DATA>
L4_uint64_t L4virtio::Svr::Driver_mem_region_t< DATA >::drv_base () const [inline]
```

**Returns**

The base address used by the driver.

Definition at line 495 of file [l4virtio](#).

**14.335.3.3 ds()**

```
template<typename DATA>
L4::Cap<L4Re::Dataspace> L4virtio::Svr::Driver_mem_region_t< DATA >::ds () const [inline]
```

**Returns**

The data space capability for this region.

Definition at line 507 of file [l4virtio](#).

**14.335.3.4 ds\_offset()**

```
template<typename DATA>
L4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::ds_offset () const [inline]
```

**Returns**

The offset within the data space.

Definition at line 504 of file [l4virtio](#).

#### 14.335.3.5 empty()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::empty () const [inline]
```

##### Returns

True if the region is empty (size == 0), false else.

Definition at line 491 of file [l4virtio](#).

#### 14.335.3.6 flags()

```
template<typename DATA>
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags () const [inline]
```

##### Returns

The flags for this region.

Definition at line 488 of file [l4virtio](#).

#### 14.335.3.7 is\_writable()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::is_writable () const [inline]
```

##### Returns

True if the region is writable, false else.

Definition at line 485 of file [l4virtio](#).

#### 14.335.3.8 local()

```
template<typename DATA>
template<typename T >
T* L4virtio::Svr::Driver_mem_region_t< DATA >::local (
 Ptr< T > p) const [inline]
```

Get the local address for driver address *p*.

## Parameters

|          |                              |
|----------|------------------------------|
| <i>p</i> | Driver address to translate. |
|----------|------------------------------|

## Precondition

*p must* be contained in this region.

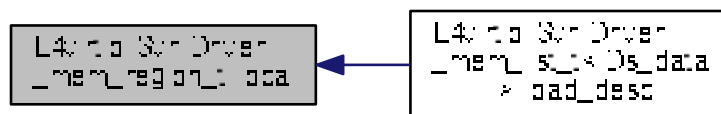
## Returns

Local address for the given driver address *p*.

Definition at line 540 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< Ds\\_data >::load\\_desc\(\)](#).

Here is the caller graph for this function:



## 14.335.3.9 local\_base()

```
template<typename DATA>
void* L4virtio::Svr::Driver_mem_region_t< DATA >::local_base () const [inline]
```

## Returns

The local base address.

Definition at line 498 of file [l4virtio](#).

### 14.335.3.10 size()

```
template<typename DATA>
l4_umword_t L4virtio::Svr::Driver_mem_region_t< DATA >::size () const [inline]
```

#### Returns

The size of the region in bytes.

Definition at line 501 of file [l4virtio](#).

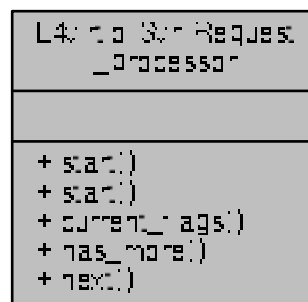
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/l4virtio](#)

## 14.336 L4virtio::Svr::Request\_processor Class Reference

Encapsulate the state for processing a VIRTIO request.

Collaboration diagram for L4virtio::Svr::Request\_processor:



### Public Member Functions

- template<typename DESC\_MAN , typename ... ARGS>  
void [start](#) (DESC\_MAN \*dm, [Virtqueue](#) \*ring, [Virtqueue::Head\\_desc](#) const &request, ARGS... args)  
*Start processing a new request.*
- template<typename DESC\_MAN , typename ... ARGS>  
[Virtqueue::Request](#) const & [start](#) (DESC\_MAN \*dm, [Virtqueue::Request](#) const &request, ARGS... args)  
*Start processing a new request.*
- [Virtqueue::Desc::Flags](#) [current\\_flags](#) () const  
*Get the flags of the currently processed descriptor.*
- bool [has\\_more](#) () const  
*Are there more chained descriptors ?*
- template<typename DESC\_MAN , typename ... ARGS>  
bool [next](#) (DESC\_MAN \*dm, ARGS... args)  
*Switch to the next descriptor in a descriptor chain.*

### 14.336.1 Detailed Description

Encapsulate the state for processing a VIRTIO request.

A VIRTIO request is a possibly chained list of descriptors retrieved from the available ring of a virtqueue, using [Virtqueue::next\\_avail\(\)](#).

The descriptor processing depends on helper (DESC\_MAN) for interpreting the descriptors in the context of the device implementation.

DESC\_MAN has to provide the functionality to safely dereference a descriptor from a descriptor list.

The following methods must be provided by DESC\_MAN:

- `DESC_MAN::load_desc(Virtqueue::Desc const &desc,  
Request_processor const *proc,  
Virtqueue::Desc const **table)`

This function is used to dereference *desc* as an indirect descriptor table, and must return a pointer to an indirect descriptor table.

- `DESC_MAN::load_desc(Virtqueue::Desc const &desc,  
Request_processor const *proc, ...)`

This function is used to dereference a descriptor as a normal data buffer, and '...' are the arguments that are passed to [start\(\)](#) and [next\(\)](#).

Definition at line [405](#) of file [virtio](#).

### 14.336.2 Member Function Documentation

#### 14.336.2.1 current\_flags()

```
Virtqueue::Desc::Flags L4virtio::Svr::Request_processor::current_flags () const [inline]
```

Get the flags of the currently processed descriptor.

#### Returns

The flags of the currently processed descriptor.

Definition at line [475](#) of file [virtio](#).

References [L4virtio::Virtqueue::Desc::flags](#).

### 14.336.2.2 has\_more()

```
bool L4virtio::Svr::Request_processor::has_more () const [inline]
```

Are there more chained descriptors ?

#### Returns

true if there are more chained descriptors in the current request.

Definition at line 482 of file [virtio](#).

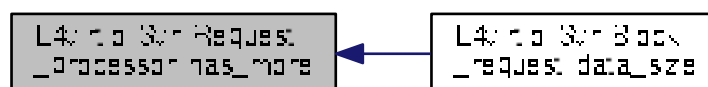
References [L4virtio::Virtqueue::Desc::flags](#), and [L4virtio::Virtqueue::Desc::Flags::next\(\)](#).

Referenced by [L4virtio::Svr::Block\\_request< Ds\\_data >::data\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.336.2.3 next()

```
template<typename DESC_MAN , typename ... ARGS>
bool L4virtio::Svr::Request_processor::next (
 DESC_MAN * dm,
 ARGS... args) [inline]
```

Switch to the next descriptor in a descriptor chain.



## Template Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <i>DESCM_MAN</i> | Type of descriptor manager (implicit). |
|------------------|----------------------------------------|

## Parameters

|             |                                                                           |
|-------------|---------------------------------------------------------------------------|
| <i>dm</i>   | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>args</i> | Extra arguments passed to dm->load_desc()                                 |

## Return values

|              |                                 |
|--------------|---------------------------------|
| <i>true</i>  | A next descriptor is available. |
| <i>false</i> | No descriptor available.        |

## Exceptions

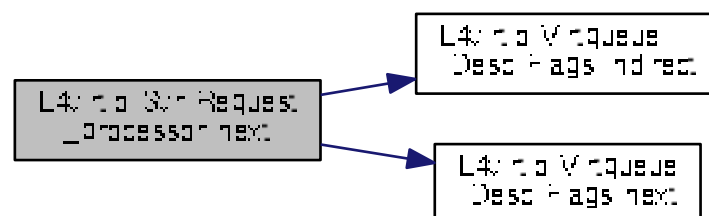
|                                       |                                                            |
|---------------------------------------|------------------------------------------------------------|
| <a href="#"><i>Bad_descriptor</i></a> | The <code>next</code> index of this descriptor is invalid. |
|---------------------------------------|------------------------------------------------------------|

Definition at line 499 of file [virtio](#).

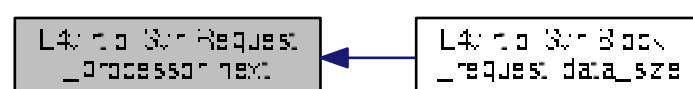
References [L4virtio::Svr::Bad\\_descriptor::Bad\\_flags](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_next](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), and [L4virtio::Virtqueue::Desc::next](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**14.336.2.4 start()** [1/2]

```
template<typename DESC_MAN , typename ... ARGS>
void L4virtio::Svr::Request_processor::start (
 DESC_MAN * dm,
 Virtqueue * ring,
 Virtqueue::Head_desc const & request,
 ARGS... args) [inline]
```

Start processing a new request.

**Template Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <i>DESCM_MAN</i> | Type of descriptor manager (implicit). |
|------------------|----------------------------------------|

**Parameters**

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| <i>dm</i>      | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>ring</i>    | VIRTIO ring of the request.                                               |
| <i>request</i> | VIRTIO request from <a href="#">Virtqueue::next_avail()</a>               |
| <i>args</i>    | Extra arguments passed to dm->load_desc()                                 |

**Precondition**

The given request must be valid.

**Exceptions**

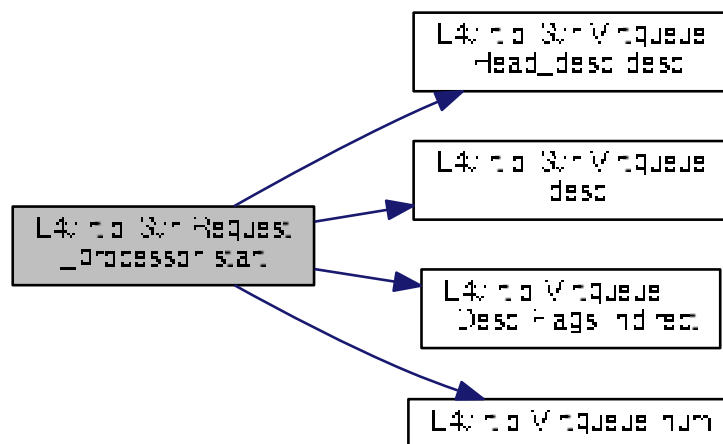
|                                |                                                                                      |
|--------------------------------|--------------------------------------------------------------------------------------|
| <a href="#">Bad_descriptor</a> | The descriptor has an invalid size or load_desc() has thrown an exception by itself. |
|--------------------------------|--------------------------------------------------------------------------------------|

Definition at line [434](#) of file [virtio](#).

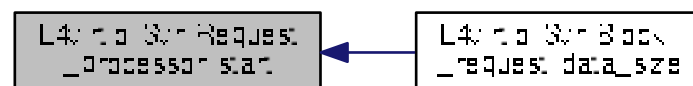
References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), [L4virtio::Svr::Virtqueue::Head\\_desc::desc\(\)](#), [L4virtio::Svr::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Virtqueue::Desc::len](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.336.2.5 start() [2/2]

```

template<typename DESC_MAN , typename ... ARGS>
Virtqueue::Request const& L4virtio::Svr::Request_processor::start (
 DESC_MAN * dm,
 Virtqueue::Request const & request,
 ARGS... args) [inline]

```

Start processing a new request.

#### Template Parameters

|                      |                                        |
|----------------------|----------------------------------------|
| <code>DESCMAN</code> | Type of descriptor manager (implicit). |
|----------------------|----------------------------------------|

**Parameters**

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| <i>dm</i>      | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>request</i> | VIRTIO request from <a href="#">Virtqueue::next_avail()</a>               |
| <i>args</i>    | Extra arguments passed to dm->load_desc()                                 |

**Precondition**

The given request must be valid.

Definition at line [465](#) of file [virtio](#).

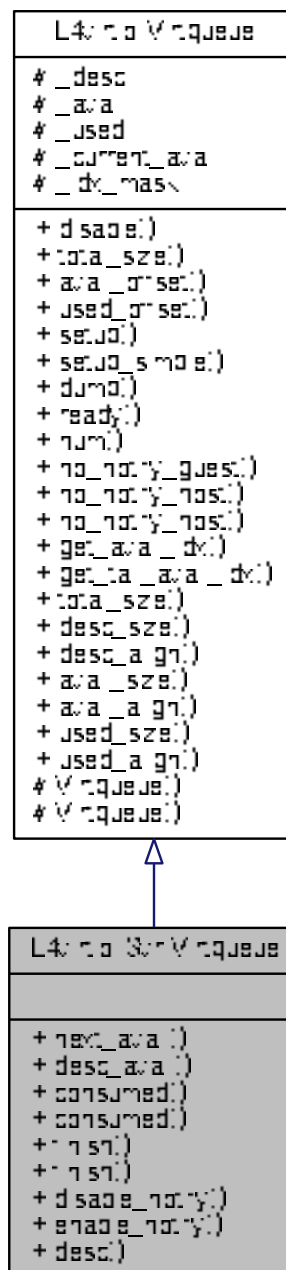
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/virtio](#)

## 14.337 L4virtio::Svr::Virtqueue Class Reference

[Virtqueue](#) implementation for the device.

Inheritance diagram for L4virtio::Svr::Virtqueue:





## Public Member Functions

- Request [next\\_avail](#) ()  
*Get the next available descriptor from the available ring.*
- bool [desc\\_avail](#) () const  
*Test for available descriptors.*
- void [consumed](#) ([Head\\_desc](#) const &*r*, [l4\\_uint32\\_t](#) *len*=0)  
*Put the given descriptor into the used ring.*
- void [disable\\_notify](#) ()  
*Set the 'no notify' flag for this queue.*
- void [enable\\_notify](#) ()  
*Clear the 'no notify' flag for this queue.*
- [Desc](#) const \* [desc](#) (unsigned *idx*) const  
*Get a descriptor from the descriptor list.*

## Additional Inherited Members

### 14.337.1 Detailed Description

[Virtqueue](#) implementation for the device.

This class represents a single virtqueue, with a local running available index.

Definition at line 94 of file [virtio](#).

### 14.337.2 Member Function Documentation

#### 14.337.2.1 consumed()

```
void L4virtio::Svr::Virtqueue::consumed (
 Head_desc const & r,
 l4_uint32_t len = 0) [inline]
```

Put the given descriptor into the used ring.

#### Parameters

|            |                                           |
|------------|-------------------------------------------|
| <i>r</i>   | request that shall be marked as finished. |
| <i>len</i> | the total number of bytes written.        |

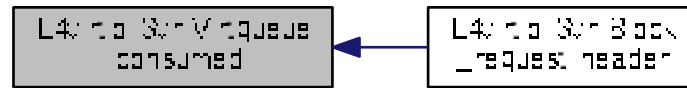
#### Precondition

queue must be in working state.  
*r* must be a valid request from this queue.

Definition at line 173 of file [virtio](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::header\(\)](#).

Here is the caller graph for this function:



#### 14.337.2.2 desc()

```
Desc const* L4virtio::Svr::Virtqueue::desc (
 unsigned idx) const [inline]
```

Get a descriptor from the descriptor list.

##### Parameters

|            |                              |
|------------|------------------------------|
| <i>idx</i> | the index of the descriptor. |
|------------|------------------------------|

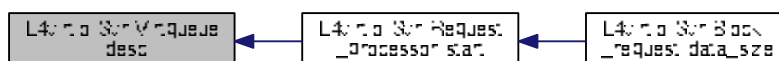
##### Precondition

$idx < num$   
queue must be in working state

Definition at line 238 of file [virtio](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:





### 14.337.2.3 desc\_avail()

```
bool L4virtio::Svr::Virtqueue::desc_avail () const [inline]
```

Test for available descriptors.

#### Returns

true if there are descriptors available, false if not.

#### Precondition

The queue must be in working state.

Definition at line 161 of file [virtio](#).

### 14.337.2.4 disable\_notify()

```
void L4virtio::Svr::Virtqueue::disable_notify () [inline]
```

Set the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 215 of file [virtio](#).

References [L4\\_LIKELY](#).

### 14.337.2.5 enable\_notify()

```
void L4virtio::Svr::Virtqueue::enable_notify () [inline]
```

Clear the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 226 of file [virtio](#).

References [L4\\_LIKELY](#).

14.337.2.6 `next_avail()`

```
Request L4virtio::Svr::Virtqueue::next_avail () [inline]
```

Get the next available descriptor from the available ring.

**Precondition**

The queue must be in working state.

**Returns**

A Request for the next available descriptor, the Request is invalid if there are no descriptors in the available ring.

**Note**

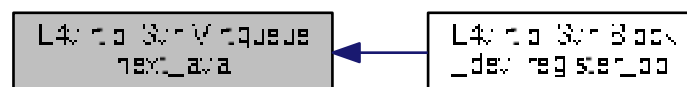
The return value must be checked even when a previous `desc_avail()` returned true.

Definition at line 144 of file [virtio](#).

References [L4\\_LIKELY](#).

Referenced by [L4virtio::Svr::Block\\_dev<Ds\\_data>::register\\_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

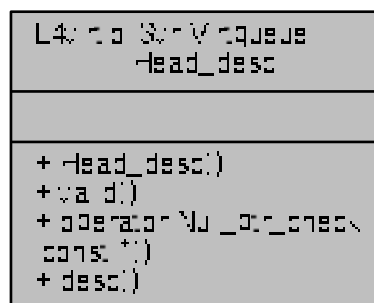
- `I4/I4virtio/server/virtio`

## 14.338 L4virtio::Svr::Virtqueue::Head\_desc Class Reference

VIRTIO request, essentially a descriptor from the available ring.

Inherited by L4virtio::Svr::Virtqueue::Request.

Collaboration diagram for L4virtio::Svr::Virtqueue::Head\_desc:



### Public Member Functions

- [Head\\_desc](#) ()  
*Make invalid (NULL) request.*
- bool [valid](#) () const
- [operator Null\\_ptr\\_check const \\*](#) () const
- [Desc](#) const \* [desc](#) () const

#### 14.338.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

Definition at line [100](#) of file [virtio](#).

#### 14.338.2 Member Function Documentation

### 14.338.2.1 desc()

```
Desc const* L4virtio::Svr::Virtqueue::Head_desc::desc () const [inline]
```

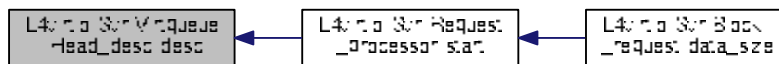
#### Returns

Pointer to the head descriptor of the request.

Definition at line 121 of file [virtio](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



### 14.338.2.2 operator Null\_ptr\_check const \*()

```
L4virtio::Svr::Virtqueue::Head_desc::operator Null_ptr_check const * () const [inline]
```

#### Returns

True if the request is valid (not NULL).

Definition at line 117 of file [virtio](#).

### 14.338.2.3 valid()

```
bool L4virtio::Svr::Virtqueue::Head_desc::valid () const [inline]
```

#### Returns

True if the request is valid (not NULL).

Definition at line 114 of file [virtio](#).

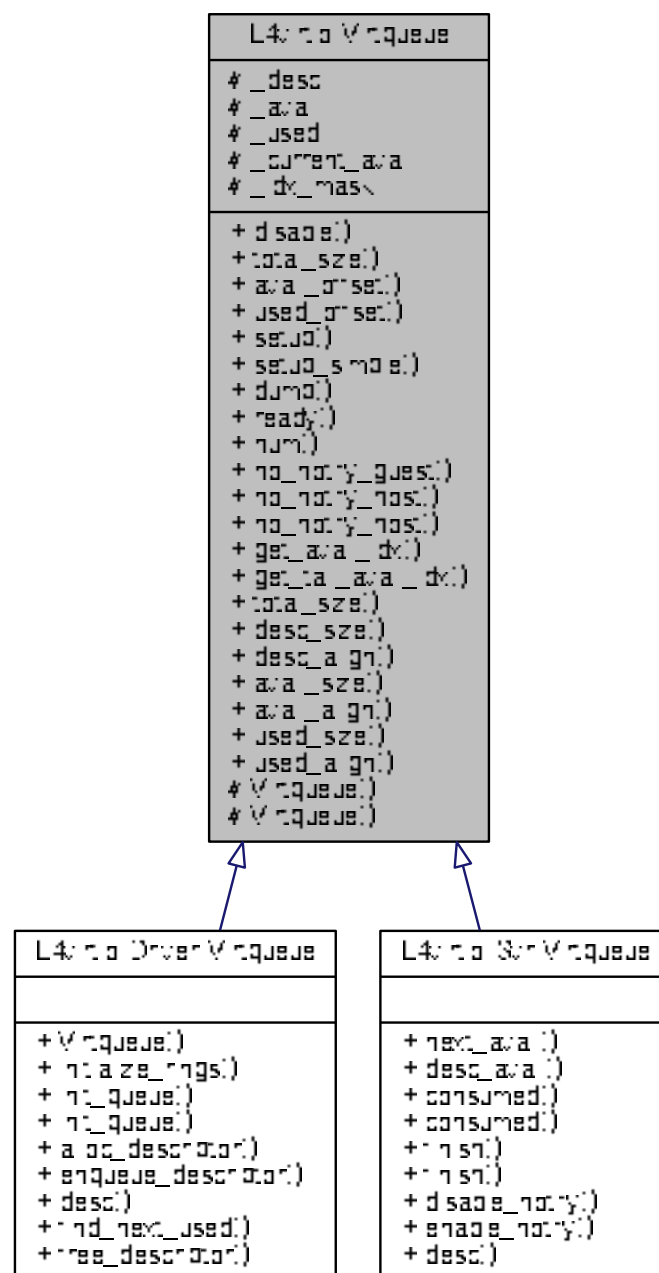
The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio`

## 14.339 L4virtio::Virtqueue Class Reference

Low-level [Virtqueue](#).

Inheritance diagram for L4virtio::Virtqueue:





## Public Types

- enum  
*Fixed alignment values for different parts of a virtqueue.*

## Public Member Functions

- void `disable` ()  
*Completely disable the queue.*
- unsigned long `total_size` () const  
*Calculate the total size of this virtqueue.*
- unsigned long `avail_offset` () const  
*Get the offset of the available ring from the descriptor table.*
- unsigned long `used_offset` () const  
*Get the offset of the used ring from the descriptor table.*
- void `setup` (unsigned `num`, void \*`desc`, void \*`avail`, void \*`used`)  
*Enable this queue.*
- void `setup_simple` (unsigned `num`, void \*`ring`)  
*Enable this queue.*
- void `dump` (Desc const \*`d`) const  
*Dump descriptors for this queue.*
- bool `ready` () const  
*Test if this queue is in working state.*
- unsigned `num` () const
- bool `no_notify_guest` () const  
*Get the no IRQ flag of this queue.*
- bool `no_notify_host` () const  
*Get the no notify flag of this queue.*
- void `no_notify_host` (bool `value`)  
*Set the no-notify flag for this queue.*
- `l4_uint16_t` `get_avail_idx` () const  
*Get available index from available ring (for debugging).*
- `l4_uint16_t` `get_tail_avail_idx` () const  
*Get tail-available index stored in local state (for debugging).*

## Static Public Member Functions

- static unsigned long `total_size` (unsigned `num`)  
*Calculate the total size for a virtqueue of the given dimensions.*
- static unsigned long `desc_size` (unsigned `num`)  
*Calculate the size of the descriptor table for `num` entries.*
- static unsigned long `desc_align` ()  
*Get the alignment in zero LSBs needed for the descriptor table.*
- static unsigned long `avail_size` (unsigned `num`)  
*Calculate the size of the available ring for `num` entries.*
- static unsigned long `avail_align` ()  
*Get the alignment in zero LSBs needed for the available ring.*
- static unsigned long `used_size` (unsigned `num`)  
*Calculate the size of the used ring for `num` entries.*
- static unsigned long `used_align` ()  
*Get the alignment in zero LSBs needed for the used ring.*

## Protected Member Functions

- [Virtqueue \(\)](#)  
*Create a disabled virtqueue.*

## Protected Attributes

- [Desc \\* \\_desc](#)  
*pointer to descriptor table, NULL if queue is off.*
- [Avail \\* \\_avail](#)  
*pointer to available ring.*
- [Used \\* \\_used](#)  
*pointer to used ring.*
- [l4\\_uint16\\_t \\_current\\_avail](#)  
*The life counter for the queue.*
- [l4\\_uint16\\_t \\_idx\\_mask](#)  
*mask used for indexing into the descriptor table and the rings.*

### 14.339.1 Detailed Description

Low-level [Virtqueue](#).

This class represents a single virtqueue, with a local running available index.

Definition at line 87 of file [virtqueue](#).

### 14.339.2 Member Function Documentation

#### 14.339.2.1 [avail\\_align\(\)](#)

```
static unsigned long L4virtio::Virtqueue::avail_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the available ring.

#### Returns

The alignment in zero LSBs needed for an available ring.

Definition at line 291 of file [virtqueue](#).

#### 14.339.2.2 [avail\\_size\(\)](#)

```
static unsigned long L4virtio::Virtqueue::avail_size (
 unsigned num) [inline], [static]
```

Calculate the size of the available ring for `num` entries.



## Parameters

|            |                                              |
|------------|----------------------------------------------|
| <i>num</i> | The number of entries in the available ring. |
|------------|----------------------------------------------|

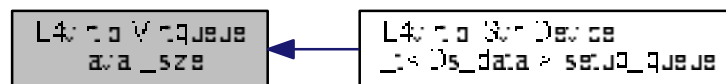
## Returns

The size in bytes needed for an available ring with *num* entries.

Definition at line 283 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.339.2.3 desc\_align()

```
static unsigned long L4virtio::Virtqueue::desc_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the descriptor table.

## Returns

The alignment in zero LSBs needed for a descriptor table.

Definition at line 273 of file [virtqueue](#).

## 14.339.2.4 desc\_size()

```
static unsigned long L4virtio::Virtqueue::desc_size (
 unsigned num) [inline], [static]
```

Calculate the size of the descriptor table for *num* entries.

## Parameters

|            |                                                |
|------------|------------------------------------------------|
| <i>num</i> | The number of entries in the descriptor table. |
|------------|------------------------------------------------|

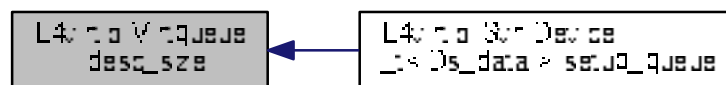
## Returns

The size in bytes needed for a descriptor table with *num* entries.

Definition at line 265 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.339.2.5 disable()

```
void L4virtio::Virtqueue::disable () [inline]
```

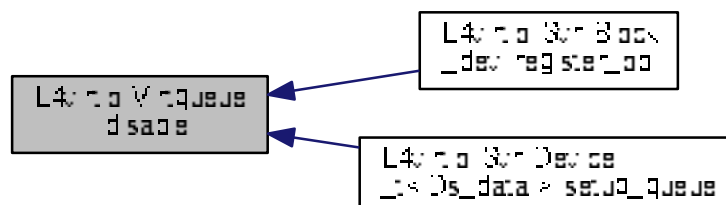
Completely disable the queue.

[setup\(\)](#) must be used to enable the queue again.

Definition at line 228 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Block\\_dev<Ds\\_data>::register\\_obj\(\)](#), and [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.339.2.6 dump()

```
void L4virtio::Virtqueue::dump (
 Desc const * d) const [inline]
```

Dump descriptors for this queue.

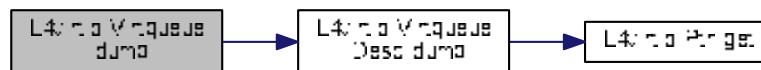
**Precondition**

the queue must be in working state.

Definition at line 395 of file [virtqueue](#).

References [L4virtio::Virtqueue::Desc::dump\(\)](#).

Here is the call graph for this function:



## 14.339.2.7 get\_avail\_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_avail_idx () const [inline]
```

Get available index from available ring (for debugging).

**Precondition**

Queue must be in a working state.

**Returns**

current index in the available ring (shared between device model and device driver).

Definition at line 452 of file [virtqueue](#).

References [L4virtio::Virtqueue::Avail::idx](#).

**14.339.2.8** `get_tail_avail_idx()`

```
l4_uint16_t L4virtio::Virtqueue::get_tail_avail_idx () const [inline]
```

Get tail-available index stored in local state (for debugging).

**Returns**

current tail index for the the available ring.

Definition at line [459](#) of file [virtqueue](#).

**14.339.2.9** `no_notify_guest()`

```
bool L4virtio::Virtqueue::no_notify_guest () const [inline]
```

Get the no IRQ flag of this queue.

**Precondition**

queue must be in working state.

**Returns**

true if the guest does not want to get IRQs (currently).

Definition at line [417](#) of file [virtqueue](#).

References [L4virtio::Virtqueue::Avail::flags](#), and [L4virtio::Virtqueue::Avail::Flags::no\\_irq\(\)](#).

Here is the call graph for this function:



## 14.339.2.10 no\_notify\_host() [1/2]

```
bool L4virtio::Virtqueue::no_notify_host () const [inline]
```

Get the no notify flag of this queue.

**Precondition**

queue must be in working state.

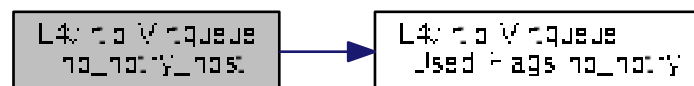
**Returns**

true if the host does not want to get IRQs (currently).

Definition at line 429 of file [virtqueue](#).

References [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Here is the call graph for this function:



## 14.339.2.11 no\_notify\_host() [2/2]

```
void L4virtio::Virtqueue::no_notify_host (
 bool value) [inline]
```

Set the no-notify flag for this queue.

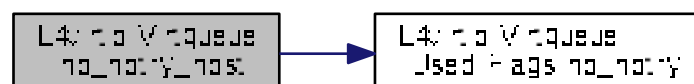
**Precondition**

Queue must be in a working state.

Definition at line 439 of file [virtqueue](#).

References [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Here is the call graph for this function:



## 14.339.2.12 num()

```
unsigned L4virtio::Virtqueue::num () const [inline]
```

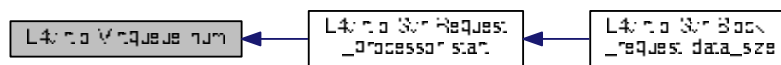
## Returns

The number of entries in the ring.

Definition at line 407 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



## 14.339.2.13 ready()

```
bool L4virtio::Virtqueue::ready () const [inline]
```

Test if this queue is in working state.

## Returns

true when the queue is in working state, false else.

Definition at line 403 of file [virtqueue](#).

References [L4\\_LIKELY](#).

Referenced by [L4virtio::Svr::Block\\_dev<Ds\\_data>::register\\_obj\(\)](#).

Here is the caller graph for this function:



14.339.2.14 `setup()`

```
void L4virtio::Virtqueue::setup (
 unsigned num,
 void * desc,
 void * avail,
 void * used) [inline]
```

Enable this queue.

## Parameters

|              |                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------|
| <i>num</i>   | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
| <i>desc</i>  | The address of the descriptor table. (Must be Desc_align aligned and at least desc_size(num) bytes in size.) |
| <i>avail</i> | The address of the available ring. (Must be Avail_align aligned and at least avail_size(num) bytes in size.) |
| <i>used</i>  | The address of the used ring. (Must be Used_align aligned and at least used_size(num) bytes in size.)        |

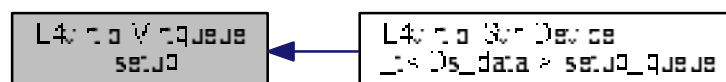
Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 353 of file [virtqueue](#).

References [L4\\_EINVAL](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:

14.339.2.15 `setup_simple()`

```
void L4virtio::Virtqueue::setup_simple (
 unsigned num,
 void * ring) [inline]
```

Enable this queue.

## Parameters

|             |                                                                                                                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>num</i>  | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).                                                                                                    |
| <i>ring</i> | The base address for the queue data structure. The memory block at <i>ring</i> must be at least <code>total_size(num)</code> bytes in size and have an alignment of <code>Desc_align(desc_align())</code> bits. |

Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 382 of file [virtqueue](#).

References [l4\\_round\\_size\(\)](#).

Here is the call graph for this function:

14.339.2.16 `total_size()` [1/2]

```
static unsigned long L4virtio::Virtqueue::total_size (
 unsigned num) [inline], [static]
```

Calculate the total size for a virtqueue of the given dimensions.

## Parameters

|            |                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
|------------|--------------------------------------------------------------------------------------------------------------|

## Returns

The total size in bytes of the queue data structures.

Definition at line 249 of file [virtqueue](#).

14.339.2.17 `total_size()` [2/2]

```
unsigned long L4virtio::Virtqueue::total_size () const [inline]
```

Calculate the total size of this virtqueue.



**Precondition**

The queue has been set up.

Definition at line 318 of file [virtqueue](#).

**14.339.2.18 used\_align()**

```
static unsigned long L4virtio::Virtqueue::used_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the used ring.

**Returns**

The alignment in zero LSBs needed for an used ring.

Definition at line 310 of file [virtqueue](#).

**14.339.2.19 used\_size()**

```
static unsigned long L4virtio::Virtqueue::used_size (
 unsigned num) [inline], [static]
```

Calculate the size of the used ring for `num` entries.

**Parameters**

|            |                                         |
|------------|-----------------------------------------|
| <i>num</i> | The number of entries in the used ring. |
|------------|-----------------------------------------|

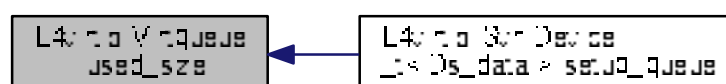
**Returns**

The size in bytes needed for an used ring with `num` entries.

Definition at line 302 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



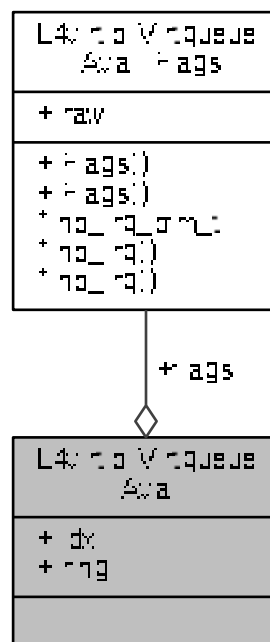
The documentation for this class was generated from the following file:

- `l4/l4virtio/virtqueue`

## 14.340 L4virtio::Virtqueue::Avail Class Reference

Type of available ring, this is read-only for the host.

Collaboration diagram for L4virtio::Virtqueue::Avail:



### Data Structures

- struct [Flags](#)  
*Flags of the available ring.*

### Data Fields

- [Flags flags](#)  
*flags of available ring*
- [l4\\_uint16\\_t idx](#)  
*available index written by guest*
- [l4\\_uint16\\_t ring \[\]](#)  
*array of available descriptor indexes.*

### 14.340.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line 134 of file [virtqueue](#).

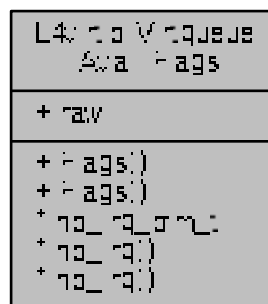
The documentation for this class was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 14.341 L4virtio::Virtqueue::Avail::Flags Struct Reference

[Flags](#) of the available ring.

Collaboration diagram for L4virtio::Virtqueue::Avail::Flags:



### Public Member Functions

- [Flags](#) ([l4\\_uint16\\_t](#) v)  
Make [Flags](#) from the raw value.

### Data Fields

- [l4\\_uint16\\_t](#) raw  
raw 16bit flags value of the available ring.
- typedef [cxx::Bitfield](#)< decltype(raw), 0, 0 > [no\\_irq\\_bfm\\_t](#)  
Guest does not want to receive interrupts when requests are finished.
- [no\\_irq\\_bfm\\_t::Val](#) no\_irq () const  
Get the no\_irq bits ( 0 to 0 ) of raw .
- [no\\_irq\\_bfm\\_t::Ref](#) no\_irq ()  
Get a reference to the no\_irq bits ( 0 to 0 ) of raw .

### 14.341.1 Detailed Description

Flags of the available ring.

Definition at line 140 of file [virtqueue](#).

### 14.341.2 Member Typedef Documentation

#### 14.341.2.1 no\_irq\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 0 , 0 > L4virtio::Virtqueue::Avail::Flags::no_irq_bfm←
_t
```

Guest does not want to receive interrupts when requests are finished.

Type to access the *no\_irq* bits ( 0 to 0 ) of *raw* .

Definition at line 149 of file [virtqueue](#).

### 14.341.3 Member Function Documentation

#### 14.341.3.1 no\_irq() [1/2]

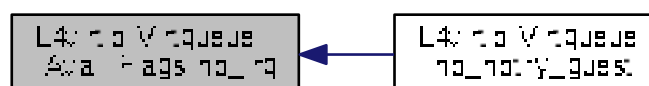
```
no_irq_bfm_t:Val L4virtio::Virtqueue::Avail::Flags::no_irq () const [inline]
```

Get the *no\_irq* bits ( 0 to 0 ) of *raw* .

Definition at line 149 of file [virtqueue](#).

Referenced by [L4virtio::Virtqueue::no\\_notify\\_guest\(\)](#).

Here is the caller graph for this function:



## 14.341.3.2 no\_irq() [2/2]

```
no_irq_bfm_t::Ref L4virtio::Virtqueue::Avail::Flags::no_irq () [inline]
```

Get a reference to the *no\_irq* bits ( 0 to 0 ) of *raw* .

Definition at line 149 of file [virtqueue](#).

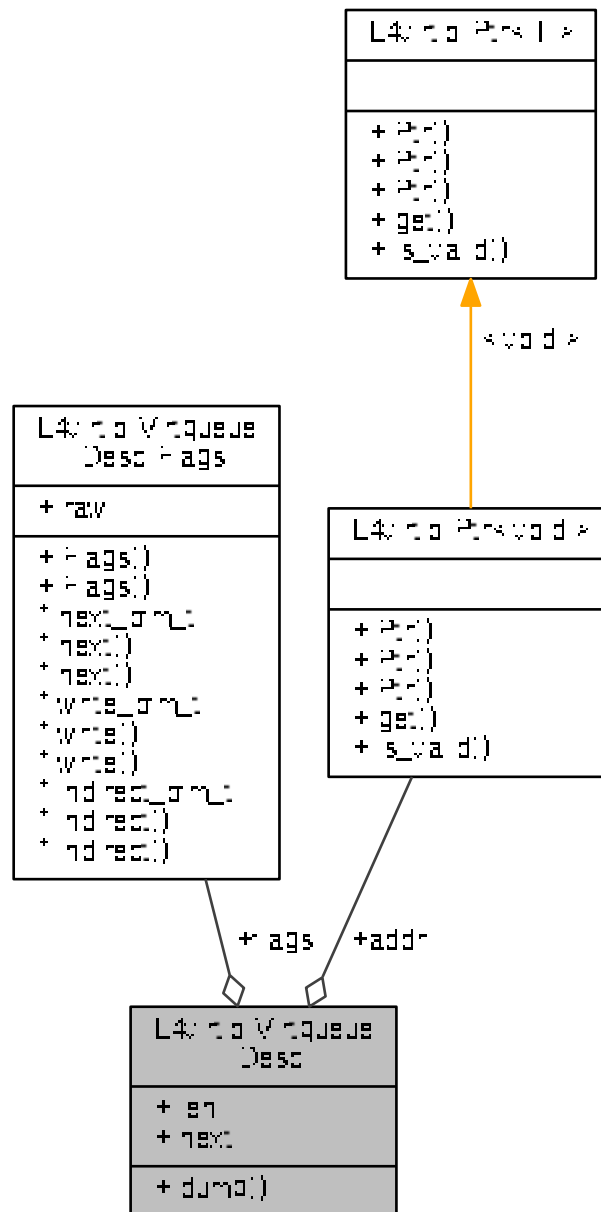
The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

## 14.342 L4virtio::Virtqueue::Desc Class Reference

Descriptor in the descriptor table.

Collaboration diagram for L4virtio::Virtqueue::Desc:



## Data Structures

- struct **Flags**  
*Type for descriptor flags.*

## Public Member Functions

- void **dump** (unsigned idx) const  
*Dump a single descriptor.*

## Data Fields

- [Ptr](#)< void > [addr](#)  
*Address stored in descriptor.*
- [l4\\_uint32\\_t](#) [len](#)  
*Length of described buffer.*
- [Flags](#) [flags](#)  
*Descriptor flags.*
- [l4\\_uint16\\_t](#) [next](#)  
*Index of the next chained descriptor.*

### 14.342.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 93 of file [virtqueue](#).

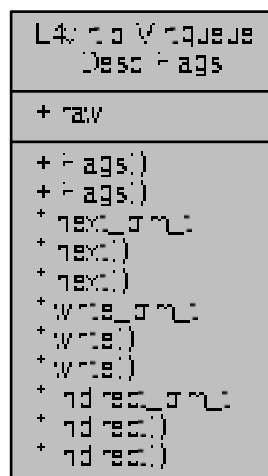
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 14.343 L4virtio::Virtqueue::Desc::Flags Struct Reference

Type for descriptor flags.

Collaboration diagram for L4virtio::Virtqueue::Desc::Flags:



## Public Member Functions

- [Flags](#) ([l4\\_uint16\\_t](#) v)  
*Make [Flags](#) from raw 16bit value.*

## Data Fields

- [l4\\_uint16\\_t](#) raw  
*raw flags value of a virtio descriptor.*
- typedef [cxx::Bitfield](#)< decltype(raw), 0, 0 > [next\\_bfm\\_t](#)  
*Part of a descriptor chain which is continued with the next field.*
- [next\\_bfm\\_t::Val](#) next () const  
*Get the next bits ( 0 to 0 ) of raw .*
- [next\\_bfm\\_t::Ref](#) next ()  
*Get a reference to the next bits ( 0 to 0 ) of raw .*
- typedef [cxx::Bitfield](#)< decltype(raw), 1, 1 > [write\\_bfm\\_t](#)  
*Block described by this descriptor is writeable.*
- [write\\_bfm\\_t::Val](#) write () const  
*Get the write bits ( 1 to 1 ) of raw .*
- [write\\_bfm\\_t::Ref](#) write ()  
*Get a reference to the write bits ( 1 to 1 ) of raw .*
- typedef [cxx::Bitfield](#)< decltype(raw), 2, 2 > [indirect\\_bfm\\_t](#)  
*Indirect descriptor, block contains a list of descriptors.*
- [indirect\\_bfm\\_t::Val](#) indirect () const  
*Get the indirect bits ( 2 to 2 ) of raw .*
- [indirect\\_bfm\\_t::Ref](#) indirect ()  
*Get a reference to the indirect bits ( 2 to 2 ) of raw .*

### 14.343.1 Detailed Description

Type for descriptor flags.

Definition at line 99 of file [virtqueue](#).

### 14.343.2 Member Typedef Documentation



## 14.343.2.1 indirect\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 2 , 2 > L4virtio::Virtqueue::Desc::Flags::indirect_bfm_t
```

Indirect descriptor, block contains a list of descriptors.

Type to access the *indirect* bits ( 2 to 2 ) of *raw* .

Definition at line 110 of file [virtqueue](#).

## 14.343.2.2 next\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 0 , 0 > L4virtio::Virtqueue::Desc::Flags::next_bfm_t
```

Part of a descriptor chain which is continued with the next field.

Type to access the *next* bits ( 0 to 0 ) of *raw* .

Definition at line 108 of file [virtqueue](#).

## 14.343.2.3 write\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 1 , 1 > L4virtio::Virtqueue::Desc::Flags::write_bfm_t
```

Block described by this descriptor is writeable.

Type to access the *write* bits ( 1 to 1 ) of *raw* .

Definition at line 108 of file [virtqueue](#).

## 14.343.3 Member Function Documentation

## 14.343.3.1 indirect() [1/2]

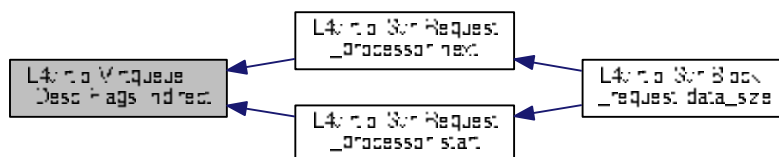
```
indirect_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::indirect () const [inline]
```

Get the *indirect* bits ( 2 to 2 ) of *raw* .

Definition at line 112 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



14.343.3.2 `indirect()` [2/2]

```
indirect_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::indirect () [inline]
```

Get a reference to the *indirect* bits ( 2 to 2 ) of *raw* .

Definition at line 112 of file [virtqueue](#).

14.343.3.3 `next()` [1/2]

```
next_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::next () [inline]
```

Get a reference to the *next* bits ( 0 to 0 ) of *raw* .

Definition at line 108 of file [virtqueue](#).

14.343.3.4 `next()` [2/2]

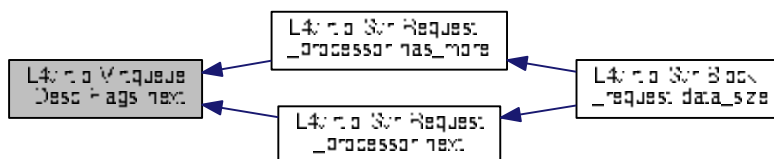
```
next_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::next () const [inline]
```

Get the *next* bits ( 0 to 0 ) of *raw* .

Definition at line 108 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), and [L4virtio::Svr::Request\\_processor::next\(\)](#).

Here is the caller graph for this function:

14.343.3.5 `write()` [1/2]

```
write_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::write () const [inline]
```

Get the *write* bits ( 1 to 1 ) of *raw* .

Definition at line 110 of file [virtqueue](#).

## 14.343.3.6 write() [2/2]

```
write_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::write () [inline]
```

Get a reference to the *write* bits ( 1 to 1 ) of *raw* .

Definition at line 110 of file [virtqueue](#).

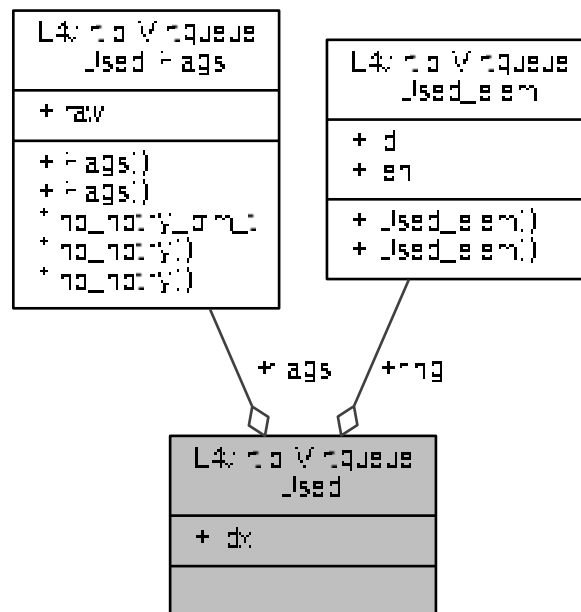
The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 14.344 L4virtio::Virtqueue::Used Class Reference

[Used](#) ring.

Collaboration diagram for L4virtio::Virtqueue::Used:



## Data Structures

- struct [Flags](#)  
*flags for the used ring.*

## Data Fields

- [Flags flags](#)  
*flags of the used ring.*
- [l4\\_uint16\\_t idx](#)  
*index of the last entry in the ring.*
- [Used\\_elem ring \[\]](#)  
*array of used descriptors.*

### 14.344.1 Detailed Description

[Used](#) ring.

Definition at line 179 of file [virtqueue](#).

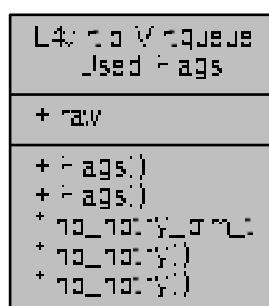
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

### 14.345 L4virtio::Virtqueue::Used::Flags Struct Reference

flags for the used ring.

Collaboration diagram for L4virtio::Virtqueue::Used::Flags:



## Public Member Functions

- [Flags \(l4\\_uint16\\_t v\)](#)  
*make [Flags](#) from raw value*

## Data Fields

- [l4\\_uint16\\_t](#) `raw`  
*raw flags value as specified by virtio.*
- typedef [cxx::Bitfield](#)< [decltype](#)(`raw`), 0, 0 > [no\\_notify\\_bfm\\_t](#)  
*host does not want to be notified when new requests have been queued.*
- [no\\_notify\\_bfm\\_t::Val](#) `no_notify` () const  
*Get the no\_notify bits ( 0 to 0 ) of raw .*
- [no\\_notify\\_bfm\\_t::Ref](#) `no_notify` ()  
*Get a reference to the no\_notify bits ( 0 to 0 ) of raw .*

### 14.345.1 Detailed Description

flags for the used ring.

Definition at line [185](#) of file [virtqueue](#).

### 14.345.2 Member Typedef Documentation

#### 14.345.2.1 no\_notify\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 0 , 0 > L4virtio::Virtqueue::Used::Flags::no_notify_↵
bfm_t
```

host does not want to be notified when new requests have been queued.

Type to access the *no\_notify* bits ( 0 to 0 ) of *raw* .

Definition at line [194](#) of file [virtqueue](#).

### 14.345.3 Member Function Documentation

14.345.3.1 `no_notify()` [1/2]

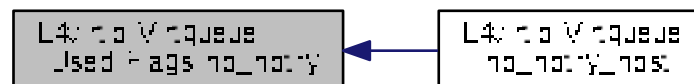
```
no_notify_bfm_t::Val L4virtio::Virtqueue::Used::Flags::no_notify () const [inline]
```

Get the *no\_notify* bits ( 0 to 0 ) of *raw* .

Definition at line 194 of file `virtqueue`.

Referenced by `L4virtio::Virtqueue::no_notify_host()`.

Here is the caller graph for this function:

14.345.3.2 `no_notify()` [2/2]

```
no_notify_bfm_t::Ref L4virtio::Virtqueue::Used::Flags::no_notify () [inline]
```

Get a reference to the *no\_notify* bits ( 0 to 0 ) of *raw* .

Definition at line 194 of file `virtqueue`.

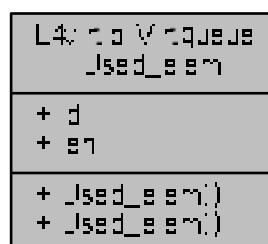
The documentation for this struct was generated from the following file:

- `l4/l4virtio/virtqueue`

14.346 `L4virtio::Virtqueue::Used_elem` Struct Reference

Type of an element of the used ring.

Collaboration diagram for `L4virtio::Virtqueue::Used_elem`:



## Public Member Functions

- [Used\\_elem](#) ([l4\\_uint16\\_t](#) id, [l4\\_uint32\\_t](#) len)  
*Initialize a used ring element.*

## Data Fields

- [l4\\_uint32\\_t](#) id  
*descriptor index*
- [l4\\_uint32\\_t](#) len  
*length field*

### 14.346.1 Detailed Description

Type of an element of the used ring.

Definition at line 160 of file [virtqueue](#).

### 14.346.2 Constructor & Destructor Documentation

#### 14.346.2.1 Used\_elem()

```
L4virtio::Virtqueue::Used_elem::Used_elem (
 l4_uint16_t id,
 l4_uint32_t len) [inline]
```

Initialize a used ring element.

#### Parameters

|            |                                                                  |
|------------|------------------------------------------------------------------|
| <i>id</i>  | The index of the descriptor to be marked as used.                |
| <i>len</i> | The total bytes written into the buffer of the descriptor chain. |

Definition at line 171 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

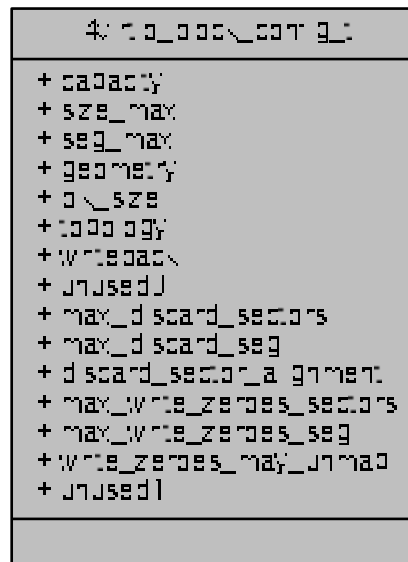
- l4/l4virtio/virtqueue

## 14.347 l4virtio\_block\_config\_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for `l4virtio_block_config_t`:



## Data Fields

- [l4\\_uint64\\_t capacity](#)  
*Capacity of device in 512-byte sectors.*
- [l4\\_uint32\\_t size\\_max](#)  
*Maximum size of a single segment.*
- [l4\\_uint32\\_t seg\\_max](#)  
*Maximum number of segments per request.*
- [l4\\_uint32\\_t blk\\_size](#)  
*Block size of underlying disk.*

### 14.347.1 Detailed Description

Device configuration for block devices.

Definition at line 80 of file [virtio\\_block.h](#).

### 14.347.2 Field Documentation



### 14.347.2.1 blk\_size

```
l4_uint32_t l4virtio_block_config_t::blk_size
```

Block size of underlying disk.

Definition at line 91 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

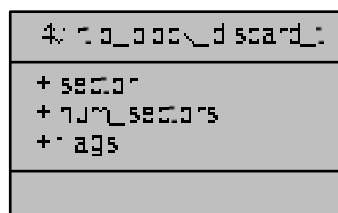
- l4/l4virtio/virtio\_block.h

## 14.348 l4virtio\_block\_discard\_t Struct Reference

Structure used for the write zeroes and discard commands.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_discard\_t:



### 14.348.1 Detailed Description

Structure used for the write zeroes and discard commands.

Definition at line 70 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

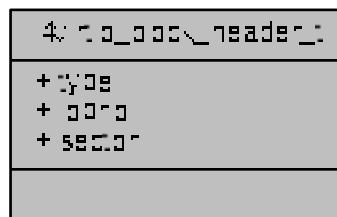
- l4/l4virtio/virtio\_block.h

## 14.349 l4virtio\_block\_header\_t Struct Reference

Header structure of a request for a block device.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_header\_t:



### Data Fields

- [l4\\_uint32\\_t type](#)  
*Kind of request, see L4virtio\_block\_operations.*
- [l4\\_uint32\\_t ioprio](#)  
*Priority (unused)*
- [l4\\_uint64\\_t sector](#)  
*First sector to read/write.*

### 14.349.1 Detailed Description

Header structure of a request for a block device.

Definition at line 54 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtio\\_block.h](#)

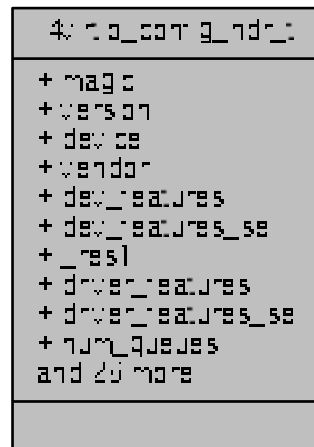
## 14.350 l4virtio\_config\_hdr\_t Struct Reference

L4-VIRTIO config header, provided in shared data space.

```
#include <virtio.h>
```

Inherited by L4virtio::Device::Config\_hdr.

Collaboration diagram for l4virtio\_config\_hdr\_t:



### Data Fields

- [l4\\_uint32\\_t magic](#)  
*magic value (must be 'virt').*
- [l4\\_uint32\\_t version](#)  
*VIRTIO version.*
- [l4\\_uint32\\_t device](#)  
*device ID*
- [l4\\_uint32\\_t vendor](#)  
*vendor ID*
- [l4\\_uint32\\_t dev\\_features](#)  
*device features windows selected by device\_feature\_sel*
- [l4\\_uint32\\_t num\\_queues](#)  
*number of virtqueues*
- [l4\\_uint32\\_t queues\\_offset](#)  
*offset of virtqueue config array*
- [l4\\_uint32\\_t status](#)  
*Device status register (read-only).*
- [l4\\_uint32\\_t cfg\\_driver\\_notify\\_index](#)  
*W: Event index to be used for config notifications (device to driver)*
- [l4\\_uint32\\_t cfg\\_device\\_notify\\_index](#)  
*R: Event index to be used for config notifications (driver to device)*
- [l4\\_uint32\\_t cmd](#)  
*L4 specific command register polled by the driver iff supported.*

### 14.350.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 127 of file [virtio.h](#).

### 14.350.2 Field Documentation

#### 14.350.2.1 magic

```
l4_uint32_t l4virtio_config_hdr_t::magic
```

magic value (must be 'virt').

Definition at line 129 of file [virtio.h](#).

#### 14.350.2.2 status

```
l4_uint32_t l4virtio_config_hdr_t::status
```

Device status register (read-only).

The register must be written using [l4virtio\\_set\\_status\(\)](#).

must be at offset 0x70 (virtio-mmio)

Definition at line 166 of file [virtio.h](#).

Referenced by [l4virtio\\_get\\_feature\(\)](#), [L4virtio::Svr::Dev\\_config::set\\_failed\(\)](#), and [L4virtio::Svr::Dev\\_config::set\\_status\(\)](#).

The documentation for this struct was generated from the following file:

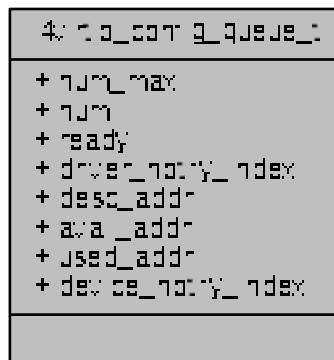
- [l4/l4virtio/virtio.h](#)

## 14.351 l4virtio\_config\_queue\_t Struct Reference

Queue configuration entry.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio\_config\_queue\_t:



### Data Fields

- [l4\\_uint16\\_t num\\_max](#)  
*R: maximum number of descriptors supported by this queue.*
- [l4\\_uint16\\_t num](#)  
*RW: number of descriptors configured for this queue.*
- [l4\\_uint16\\_t ready](#)  
*RW: queue ready flag (read-write)*
- [l4\\_uint16\\_t driver\\_notify\\_index](#)  
*W: Event index to be used for device notifications (device to driver)*
- [l4\\_uint64\\_t desc\\_addr](#)  
*W: address of descriptor table.*
- [l4\\_uint64\\_t avail\\_addr](#)  
*W: address of available ring.*
- [l4\\_uint64\\_t used\\_addr](#)  
*W: address of used ring.*
- [l4\\_uint16\\_t device\\_notify\\_index](#)  
*R: Event index to be used by the driver (driver to device)*

### 14.351.1 Detailed Description

Queue configuration entry.

An array of such entries is available at the [l4virtio\\_config\\_hdr\\_t::queues\\_offset](#) in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by `L4VIRTIO_CMD_CFG_QUEUE`.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful `L4VIRTIO_CMD_CFG_QUEUE` or after a `config_queue` IPC. Note, the change might also have immediate effect, depending on the device implementation.

Definition at line 208 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

- `l4/l4virtio/virtio.h`

## File Documentation

APIC for AMD64.

APIC for AMD64.

Definition in file [apic.h](#).

## 15.2 apic.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef __L4_UTIL_APIC_H
00014 #define __L4_UTIL_APIC_H
00015
00016 /*
00017 * Local APIC programming library
00018 *
00019 * For documentation, see
00020 *
00021 * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
00022 * "Advanced Programmable Interrupt Controller (APIC)"
00023 *
00024 * Local APIC is present since
00025 * - INTEL P6 (PPro)
00026 * - AMD K7 (Athlon), Model 2
00027 *
00028 * In non-SMP-boards, local APIC is disabled, but
00029 * can be activated by writing to a MSR register.
00030 * For using APIC see packets cpufreq and l4rtl.
00031 *
00032 * See linux/include/asm-i386/i82489.h for further details.
00033 */
00034
00035 #define APIC_PHYS_BASE 0xFEE00000
00036 #define APIC_MAP_BASE 0xA0200000
00037 #define APIC_BASE_MSR 0x1b
00038
00039 #define APIC_ID 0x20
00040 #define GET_APIC_ID(x) ((x)>>24)&0x0F)
00041 #define APIC_LVR 0x30
00042 #define GET_APIC_VERSION(x) ((x)&0xFF)
00043 #define APIC_TASKPRI 0x80
00044 #define APIC_TPRI_MASK 0xFF
00045 #define APIC_EOI 0xB0
00046 #define APIC_LDR 0xD0
00047 #define APIC_LDR_MASK (0xFF<<24)
00048 #define APIC_DFR 0xE0
00049 #define SET_APIC_DFR(x) ((x)<<28)
00050 #define APIC_SPIV 0xF0
00051 #define APIC_LVTT 0x320
00052 #define APIC_LVTPC 0x340
00053 #define APIC_LVT0 0x350
00054 #define SET_APIC_TIMER_BASE(x) ((x)<<18)
00055 #define APIC_TIMER_BASE_DIV 0x2
00056 #define APIC_LVT1 0x360
00057 #define APIC_LVTERR 0x370
00058 #define APIC_TMICT 0x380
00059 #define APIC_TMCCT 0x390
00060 #define APIC_TDCR 0x3E0
00061
00062 #define APIC_LVT_MASKED (1<<16)
00063 #define APIC_LVT_TIMER_PERIODIC (1<<17)
00064 #define APIC_TDR_DIV_1 0xB
00065 #define APIC_TDR_DIV_2 0x0
00066 #define APIC_TDR_DIV_4 0x1
00067 #define APIC_TDR_DIV_8 0x2
00068 #define APIC_TDR_DIV_16 0x3
00069 #define APIC_TDR_DIV_32 0x8
00070 #define APIC_TDR_DIV_64 0x9
00071 #define APIC_TDR_DIV_128 0xA
00072
00073 #include <l4/sys/compiler.h>
00074 #include <l4/sys/types.h>
00075
00076 EXTERN_C_BEGIN
00077
00078 /* prototypes */
00079 extern unsigned long apic_map_base;
00080 extern unsigned long apic_timer_divisor;
00081
00082 extern unsigned long l4_scaler_apic_to_ms;
00083
00084 L4_CV void apic_show_registers(void);
00085 L4_CV int apic_check_working(void);
00086 L4_CV void apic_activate_by_io(void);
00087 L4_CV void apic_timer_set_divisor(int divisor);

```



```

00088
00089 L4_CV unsigned long l4_calibrate_apic(void);
00090
00091 EXTERN_C_END
00092
00093 L4_INLINE void apic_write(unsigned long reg, unsigned long v);
00094 L4_INLINE unsigned long apic_read(unsigned long reg);
00095 L4_INLINE void apic_activate_by_msr(void);
00096 L4_INLINE void apic_deactivate_by_msr(void);
00097 L4_INLINE unsigned long apic_read_phys_address(void);
00098 L4_INLINE int apic_test_present(void);
00099 L4_INLINE void apic_soft_enable(void);
00100 L4_INLINE void apic_init(unsigned long map_addr);
00101 L4_INLINE void apic_done(void);
00102 L4_INLINE void apic_irq_ack(void);
00103
00104 L4_INLINE void apic_lvt0_disable_irq(void);
00105 L4_INLINE void apic_lvt0_enable_irq(void);
00106 L4_INLINE void apic_lvt1_disable_irq(void);
00107 L4_INLINE void apic_lvt1_enable_irq(void);
00108
00109 L4_INLINE void apic_timer_write(unsigned long value);
00110 L4_INLINE unsigned long apic_timer_read(void);
00111 L4_INLINE void apic_timer_disable_irq(void);
00112 L4_INLINE void apic_timer_enable_irq(void);
00113 L4_INLINE void apic_timer_assign_irq(unsigned long vector);
00114 L4_INLINE void apic_timer_set_periodic(void);
00115 L4_INLINE void apic_timer_set_one_shot(void);
00116
00117 L4_INLINE void apic_perf_disable_irq(void);
00118 L4_INLINE void apic_perf_enable_irq(void);
00119 L4_INLINE void apic_perf_assign_irq(unsigned long vector);
00120
00121
00122 /* write APIC register */
00123 L4_INLINE void
00124 apic_write(unsigned long reg, unsigned long v)
00125 {
00126 *((volatile unsigned long *) (apic_map_base+reg))=v;
00127 }
00128
00129
00130 /* read APIC register */
00131 L4_INLINE unsigned long
00132 apic_read(unsigned long reg)
00133 {
00134 return *((volatile unsigned long *) (apic_map_base+reg));
00135 }
00136
00137
00138 /* disable LINT0 */
00139 L4_INLINE void
00140 apic_lvt0_disable_irq(void)
00141 {
00142 unsigned long tmp_val;
00143 tmp_val = apic_read(APIC_LVT0);
00144 tmp_val |= APIC_LVT_MASKED;
00145 apic_write(APIC_LVT0, tmp_val);
00146 }
00147
00148
00149 /* enable LINT0 */
00150 L4_INLINE void
00151 apic_lvt0_enable_irq(void)
00152 {
00153 unsigned long tmp_val;
00154 tmp_val = apic_read(APIC_LVT0);
00155 tmp_val &= ~APIC_LVT_MASKED;
00156 apic_write(APIC_LVT0, tmp_val);
00157 }
00158
00159
00160 /* disable LINT1 */
00161 L4_INLINE void
00162 apic_lvt1_disable_irq(void)
00163 {
00164 unsigned long tmp_val;
00165 tmp_val = apic_read(APIC_LVT1);
00166 tmp_val |= APIC_LVT_MASKED;
00167 apic_write(APIC_LVT1, tmp_val);
00168 }
00169
00170
00171 /* enable LINT1 */
00172 L4_INLINE void
00173 apic_lvt1_enable_irq(void)
00174 {

```

```
00175 unsigned long tmp_val;
00176 tmp_val = apic_read(APIC_LVT1);
00177 tmp_val &= ~(APIC_LVT_MASKED);
00178 apic_write(APIC_LVT1, tmp_val);
00179 }
00180
00181
00182 /* write APIC timer register */
00183 L4_INLINE void
00184 apic_timer_write(unsigned long value)
00185 {
00186 apic_read(APIC_TMICT);
00187 apic_write(APIC_TMICT, value);
00188 }
00189
00190
00191 /* read APIC timer register */
00192 L4_INLINE unsigned long
00193 apic_timer_read(void)
00194 {
00195 return apic_read(APIC_TMCCT);
00196 }
00197
00198
00199 /* disable IRQ when APIC timer passes 0 */
00200 L4_INLINE void
00201 apic_timer_disable_irq(void)
00202 {
00203 unsigned long tmp_val;
00204 tmp_val = apic_read(APIC_LVTT);
00205 tmp_val |= APIC_LVT_MASKED;
00206 apic_write(APIC_LVTT, tmp_val);
00207 }
00208
00209
00210 /* enable IRQ when APIC timer passes 0 */
00211 L4_INLINE void
00212 apic_timer_enable_irq(void)
00213 {
00214 unsigned long tmp_val;
00215 tmp_val = apic_read(APIC_LVTT);
00216 tmp_val &= ~(APIC_LVT_MASKED);
00217 apic_write(APIC_LVTT, tmp_val);
00218 }
00219
00220
00221 L4_INLINE void
00222 apic_timer_set_periodic(void)
00223 {
00224 unsigned long tmp_val;
00225 tmp_val = apic_read(APIC_LVTT);
00226 tmp_val |= APIC_LVT_TIMER_PERIODIC;
00227 tmp_val |= APIC_LVT_MASKED;
00228 apic_write(APIC_LVTT, tmp_val);
00229 }
00230
00231
00232 L4_INLINE void
00233 apic_timer_set_one_shot(void)
00234 {
00235 unsigned long tmp_val;
00236 tmp_val = apic_read(APIC_LVTT);
00237 tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
00238 tmp_val |= APIC_LVT_MASKED;
00239 apic_write(APIC_LVTT, tmp_val);
00240 }
00241
00242
00243 /* set vector of APIC timer irq */
00244 L4_INLINE void
00245 apic_timer_assign_irq(unsigned long vector)
00246 {
00247 unsigned long tmp_val;
00248 tmp_val = apic_read(APIC_LVTT);
00249 tmp_val &= 0xfffffff0;
00250 tmp_val |= vector;
00251 tmp_val |= APIC_LVT_MASKED;
00252 apic_write(APIC_LVTT, tmp_val);
00253 }
00254
00255
00256 /* disable IRQ when performance counter passes 0 */
00257 L4_INLINE void
00258 apic_perf_disable_irq(void)
00259 {
00260 unsigned long tmp_val;
00261 tmp_val = apic_read(APIC_LVTPC);
```

```

00262 tmp_val |= APIC_LVT_MASKED;
00263 apic_write(APIC_LVTPC, tmp_val);
00264 }
00265
00266
00267 /* enable IRQ when performance counter passes 0 */
00268 L4_INLINE void
00269 apic_perf_enable_irq(void)
00270 {
00271 unsigned long tmp_val;
00272 tmp_val = apic_read(APIC_LVTPC);
00273 tmp_val &= ~(APIC_LVT_MASKED);
00274 apic_write(APIC_LVTPC, tmp_val);
00275 }
00276
00277
00278 /* set vector of performance counter irq */
00279 L4_INLINE void
00280 apic_perf_assign_irq(unsigned long vector)
00281 {
00282 unsigned long tmp_val;
00283 tmp_val = apic_read(APIC_LVTPC);
00284 tmp_val &= 0xffffffff00;
00285 tmp_val |= vector;
00286 tmp_val |= APIC_LVT_MASKED;
00287 apic_write(APIC_LVTPC, tmp_val);
00288 }
00289
00290
00291 /* activate APIC by writing to appropriate MSR */
00292 L4_INLINE void
00293 apic_activate_by_msr(void)
00294 {
00295 unsigned long low;
00296 unsigned long high;
00297
00298 /* rdmsr */
00299 asm volatile(".byte 0xf; .byte 0x32\n"
00300 : "=a" (low),
00301 " =d" (high)
00302 : "c" (APIC_BASE_MSR)
00303);
00304
00305 low |= 0x800; /* activate APIC */
00306 low &= 0x00000fff;
00307 low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */
00308
00309 /* wrmsr */
00310 asm volatile(".byte 0xf; .byte 0x30\n"
00311 :
00312 : "c" (APIC_BASE_MSR),
00313 "a" (low),
00314 "d" (high)
00315);
00316 }
00317
00318
00319 /* deactivate APIC by writing to appropriate MSR */
00320 L4_INLINE void
00321 apic_deactivate_by_msr(void)
00322 {
00323 unsigned long low;
00324 unsigned long high;
00325
00326 /* rdmsr */
00327 asm volatile(".byte 0xf; .byte 0x32\n"
00328 : "=a" (low),
00329 " =d" (high)
00330 : "c" (APIC_BASE_MSR)
00331);
00332
00333 low &= 0xfffff7ff; /* deactivate APIC */
00334
00335 /* wrmsr */
00336 asm volatile(".byte 0xf; .byte 0x30\n"
00337 :
00338 : "c" (APIC_BASE_MSR),
00339 "a" (low),
00340 "d" (high)
00341);
00342 }
00343
00344
00345 /* read memory mapped address of apic */
00346 L4_INLINE unsigned long
00347 apic_read_phys_address(void)
00348 {

```

```

00349 unsigned long low;
00350 unsigned long high;
00351
00352 /* rdmsr */
00353 asm volatile(".byte 0xf; .byte 0x32\n"
00354 : "=a" (low),
00355 : "d" (high)
00356 : "c" (APIC_BASE_MSR)
00357);
00358
00359 return (low &= 0xfffff000);
00360 }
00361
00362
00363 /* test if APIC present */
00364 L4_INLINE int
00365 apic_test_present(void)
00366 {
00367 unsigned int dummy;
00368 unsigned int capability;
00369
00370 asm volatile("pushl %%ebx ; cpuid ; popl %%ebx"
00371 : "=a" (dummy),
00372 : "c" (dummy),
00373 : "d" (capability)
00374 : "a" (0x00000001)
00375 : "cc");
00376
00377 return ((capability & 1<<9) !=0);
00378 }
00379
00380
00381 L4_INLINE void
00382 apic_soft_enable(void)
00383 {
00384 unsigned long tmp_val;
00385 tmp_val = apic_read(APIC_SPIV);
00386 tmp_val |= (1<<8); /* enable APIC */
00387 tmp_val &= ~(1<<9); /* enable Focus Processor Checking */
00388 tmp_val |= 0xff; /* Set spurious IRQ vector to 0xff */
00389 apic_write(APIC_SPIV, tmp_val);
00390 }
00391
00392
00393 L4_INLINE void
00394 apic_init(unsigned long base_addr)
00395 {
00396 apic_map_base = base_addr;
00397 }
00398
00399
00400 L4_INLINE void
00401 apic_done(void)
00402 {
00403 apic_map_base = 0;
00404 }
00405
00406
00407 L4_INLINE void
00408 apic_irq_ack(void)
00409 {
00410 apic_read(APIC_SPIV);
00411 apic_write(APIC_EOI, 0);
00412 }
00413
00414
00415 #endif /* __L4_UTIL_APIC_H */

```

## 15.3 x86/I4/util/apic.h File Reference

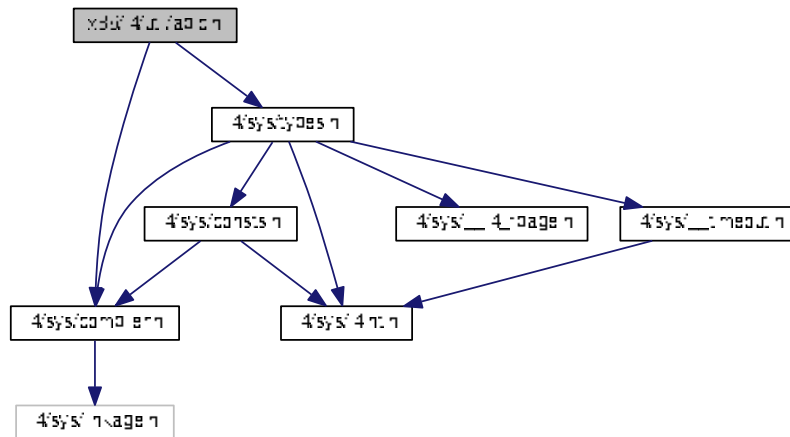
APIC for X86.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```

Include dependency graph for apic.h:



### 15.3.1 Detailed Description

APIC for X86.

Definition in file [apic.h](#).

## 15.4 apic.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef __L4_UTIL_APIC_H
00014 #define __L4_UTIL_APIC_H
00015
00016 /*
00017 * Local APIC programming library
00018 *
00019 * For documentation, see
00020 *
00021 * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
00022 * "Advanced Programmable Interrupt Controller (APIC)"
00023 *
00024 * Local APIC is present since
00025 * - INTEL P6 (PPro)
00026 * - AMD K7 (Athlon), Model 2
00027 *
00028 * In non-SMP-boards, local APIC is disabled, but
00029 * can be activated by writing to a MSR register.
00030 * For using APIC see packets cpufreq and l4rtl.
00031 *
00032 * See linux/include/asm-i386/i82489.h for further details.
00033 */
00034
00035 #define APIC_PHYS_BASE 0xFEE00000
00036 #define APIC_MAP_BASE 0xA0200000
00037 #define APIC_BASE_MSR 0x1b
00038
00039 #define APIC_ID 0x20

```

```

00040 #define GET_APIC_ID(x) ((x)>>24)&0x0F)
00041 #define APIC_LVR 0x30
00042 #define GET_APIC_VERSION(x) ((x)&0xFF)
00043 #define APIC_TASKPRI 0x80
00044 #define APIC_TPRI_MASK 0xFF
00045 #define APIC_EOI 0xB0
00046 #define APIC_LDR 0xD0
00047 #define APIC_LDR_MASK (0xFF<<24)
00048 #define APIC_DFR 0xE0
00049 #define SET_APIC_DFR(x) ((x)<<28)
00050 #define APIC_SPIV 0xF0
00051 #define APIC_LVTT 0x320
00052 #define APIC_LVTPC 0x340
00053 #define APIC_LVT0 0x350
00054 #define SET_APIC_TIMER_BASE(x) ((x)<<18)
00055 #define APIC_TIMER_BASE_DIV 0x2
00056 #define APIC_LVT1 0x360
00057 #define APIC_LVTERR 0x370
00058 #define APIC_TMICT 0x380
00059 #define APIC_TMCCT 0x390
00060 #define APIC_TDCR 0x3E0
00061
00062 #define APIC_LVT_MASKED (1<<16)
00063 #define APIC_LVT_TIMER_PERIODIC (1<<17)
00064 #define APIC_TDR_DIV_1 0xB
00065 #define APIC_TDR_DIV_2 0x0
00066 #define APIC_TDR_DIV_4 0x1
00067 #define APIC_TDR_DIV_8 0x2
00068 #define APIC_TDR_DIV_16 0x3
00069 #define APIC_TDR_DIV_32 0x8
00070 #define APIC_TDR_DIV_64 0x9
00071 #define APIC_TDR_DIV_128 0xA
00072
00073 #include <l4/sys/compiler.h>
00074 #include <l4/sys/types.h>
00075
00076 EXTERN_C_BEGIN
00077
00078 /* prototypes */
00079 extern unsigned long apic_map_base;
00080 extern unsigned long apic_timer_divisor;
00081
00082 extern unsigned long l4_scaler_apic_to_ms;
00083
00084 L4_CV void apic_show_registers(void);
00085 L4_CV int apic_check_working(void);
00086 L4_CV void apic_activate_by_io(void);
00087 L4_CV void apic_timer_set_divisor(int divisor);
00088
00089 L4_CV unsigned long l4_calibrate_apic(void);
00090
00091 EXTERN_C_END
00092
00093 L4_INLINE void apic_write(unsigned long reg, unsigned long v);
00094 L4_INLINE unsigned long apic_read(unsigned long reg);
00095 L4_INLINE void apic_activate_by_msr(void);
00096 L4_INLINE void apic_deactivate_by_msr(void);
00097 L4_INLINE unsigned long apic_read_phys_address(void);
00098 L4_INLINE int apic_test_present(void);
00099 L4_INLINE void apic_soft_enable(void);
00100 L4_INLINE void apic_init(unsigned long map_addr);
00101 L4_INLINE void apic_done(void);
00102 L4_INLINE void apic_irq_ack(void);
00103
00104 L4_INLINE void apic_lvt0_disable_irq(void);
00105 L4_INLINE void apic_lvt0_enable_irq(void);
00106 L4_INLINE void apic_lvt1_disable_irq(void);
00107 L4_INLINE void apic_lvt1_enable_irq(void);
00108
00109 L4_INLINE void apic_timer_write(unsigned long value);
00110 L4_INLINE unsigned long apic_timer_read(void);
00111 L4_INLINE void apic_timer_disable_irq(void);
00112 L4_INLINE void apic_timer_enable_irq(void);
00113 L4_INLINE void apic_timer_assign_irq(unsigned long vector);
00114 L4_INLINE void apic_timer_set_periodic(void);
00115 L4_INLINE void apic_timer_set_one_shot(void);
00116
00117 L4_INLINE void apic_perf_disable_irq(void);
00118 L4_INLINE void apic_perf_enable_irq(void);
00119 L4_INLINE void apic_perf_assign_irq(unsigned long vector);
00120
00121
00122 /* write APIC register */
00123 L4_INLINE void
00124 apic_write(unsigned long reg, unsigned long v)
00125 {
00126 *((volatile unsigned long *) (apic_map_base+reg))=v;

```

```

00127 }
00128
00129
00130 /* read APIC register */
00131 L4_INLINE unsigned long
00132 apic_read(unsigned long reg)
00133 {
00134 return *((volatile unsigned long *) (apic_map_base+reg));
00135 }
00136
00137
00138 /* disable LINT0 */
00139 L4_INLINE void
00140 apic_lvt0_disable_irq(void)
00141 {
00142 unsigned long tmp_val;
00143 tmp_val = apic_read(APIC_LVT0);
00144 tmp_val |= APIC_LVT_MASKED;
00145 apic_write(APIC_LVT0, tmp_val);
00146 }
00147
00148
00149 /* enable LINT0 */
00150 L4_INLINE void
00151 apic_lvt0_enable_irq(void)
00152 {
00153 unsigned long tmp_val;
00154 tmp_val = apic_read(APIC_LVT0);
00155 tmp_val &= ~(APIC_LVT_MASKED);
00156 apic_write(APIC_LVT0, tmp_val);
00157 }
00158
00159
00160 /* disable LINT1 */
00161 L4_INLINE void
00162 apic_lvt1_disable_irq(void)
00163 {
00164 unsigned long tmp_val;
00165 tmp_val = apic_read(APIC_LVT1);
00166 tmp_val |= APIC_LVT_MASKED;
00167 apic_write(APIC_LVT1, tmp_val);
00168 }
00169
00170
00171 /* enable LINT1 */
00172 L4_INLINE void
00173 apic_lvt1_enable_irq(void)
00174 {
00175 unsigned long tmp_val;
00176 tmp_val = apic_read(APIC_LVT1);
00177 tmp_val &= ~(APIC_LVT_MASKED);
00178 apic_write(APIC_LVT1, tmp_val);
00179 }
00180
00181
00182 /* write APIC timer register */
00183 L4_INLINE void
00184 apic_timer_write(unsigned long value)
00185 {
00186 apic_read(APIC_TMICT);
00187 apic_write(APIC_TMICT, value);
00188 }
00189
00190
00191 /* read APIC timer register */
00192 L4_INLINE unsigned long
00193 apic_timer_read(void)
00194 {
00195 return apic_read(APIC_TMCCT);
00196 }
00197
00198
00199 /* disable IRQ when APIC timer passes 0 */
00200 L4_INLINE void
00201 apic_timer_disable_irq(void)
00202 {
00203 unsigned long tmp_val;
00204 tmp_val = apic_read(APIC_LVTT);
00205 tmp_val |= APIC_LVT_MASKED;
00206 apic_write(APIC_LVTT, tmp_val);
00207 }
00208
00209
00210 /* enable IRQ when APIC timer passes 0 */
00211 L4_INLINE void
00212 apic_timer_enable_irq(void)
00213 {

```

```

00214 unsigned long tmp_val;
00215 tmp_val = apic_read(APIC_LVTT);
00216 tmp_val &= ~(APIC_LVT_MASKED);
00217 apic_write(APIC_LVTT, tmp_val);
00218 }
00219
00220
00221 L4_INLINE void
00222 apic_timer_set_periodic(void)
00223 {
00224 unsigned long tmp_val;
00225 tmp_val = apic_read(APIC_LVTT);
00226 tmp_val |= APIC_LVT_TIMER_PERIODIC;
00227 tmp_val |= APIC_LVT_MASKED;
00228 apic_write(APIC_LVTT, tmp_val);
00229 }
00230
00231
00232 L4_INLINE void
00233 apic_timer_set_one_shot(void)
00234 {
00235 unsigned long tmp_val;
00236 tmp_val = apic_read(APIC_LVTT);
00237 tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
00238 tmp_val |= APIC_LVT_MASKED;
00239 apic_write(APIC_LVTT, tmp_val);
00240 }
00241
00242
00243 /* set vector of APIC timer irq */
00244 L4_INLINE void
00245 apic_timer_assign_irq(unsigned long vector)
00246 {
00247 unsigned long tmp_val;
00248 tmp_val = apic_read(APIC_LVTT);
00249 tmp_val &= 0xffffffff00;
00250 tmp_val |= vector;
00251 tmp_val |= APIC_LVT_MASKED;
00252 apic_write(APIC_LVTT, tmp_val);
00253 }
00254
00255
00256 /* disable IRQ when performance counter passes 0 */
00257 L4_INLINE void
00258 apic_perf_disable_irq(void)
00259 {
00260 unsigned long tmp_val;
00261 tmp_val = apic_read(APIC_LVTPC);
00262 tmp_val |= APIC_LVT_MASKED;
00263 apic_write(APIC_LVTPC, tmp_val);
00264 }
00265
00266
00267 /* enable IRQ when performance counter passes 0 */
00268 L4_INLINE void
00269 apic_perf_enable_irq(void)
00270 {
00271 unsigned long tmp_val;
00272 tmp_val = apic_read(APIC_LVTPC);
00273 tmp_val &= ~(APIC_LVT_MASKED);
00274 apic_write(APIC_LVTPC, tmp_val);
00275 }
00276
00277
00278 /* set vector of performance counter irq */
00279 L4_INLINE void
00280 apic_perf_assign_irq(unsigned long vector)
00281 {
00282 unsigned long tmp_val;
00283 tmp_val = apic_read(APIC_LVTPC);
00284 tmp_val &= 0xffffffff00;
00285 tmp_val |= vector;
00286 tmp_val |= APIC_LVT_MASKED;
00287 apic_write(APIC_LVTPC, tmp_val);
00288 }
00289
00290
00291 /* activate APIC by writing to appropriate MSR */
00292 L4_INLINE void
00293 apic_activate_by_msr(void)
00294 {
00295 unsigned long low;
00296 unsigned long high;
00297
00298 /* rdmsr */
00299 asm volatile(".byte 0xf; .byte 0x32\n"
00300 : "=a" (low),

```



```

00301 "d" (high)
00302 : "c" (APIC_BASE_MSR)
00303);
00304
00305 low |= 0x800; /* activate APIC */
00306 low &= 0x00000fff;
00307 low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */
00308
00309 /* wrmsr */
00310 asm volatile(".byte 0xf; .byte 0x30\n"
00311 :
00312 : "c" (APIC_BASE_MSR),
00313 "a" (low),
00314 "d" (high)
00315);
00316 }
00317
00318
00319 /* deactivate APIC by writing to appropriate MSR */
00320 L4_INLINE void
00321 apic_deactivate_by_msr(void)
00322 {
00323 unsigned long low;
00324 unsigned long high;
00325
00326 /* rdmsr */
00327 asm volatile(".byte 0xf; .byte 0x32\n"
00328 : "=a" (low),
00329 "=d" (high)
00330 : "c" (APIC_BASE_MSR)
00331);
00332
00333 low &= 0xfffff7ff; /* deactivate APIC */
00334
00335 /* wrmsr */
00336 asm volatile(".byte 0xf; .byte 0x30\n"
00337 :
00338 : "c" (APIC_BASE_MSR),
00339 "a" (low),
00340 "d" (high)
00341);
00342 }
00343
00344
00345 /* read memory mapped address of apic */
00346 L4_INLINE unsigned long
00347 apic_read_phys_address(void)
00348 {
00349 unsigned long low;
00350 unsigned long high;
00351
00352 /* rdmsr */
00353 asm volatile(".byte 0xf; .byte 0x32\n"
00354 : "=a" (low),
00355 "=d" (high)
00356 : "c" (APIC_BASE_MSR)
00357);
00358
00359 return (low &= 0xfffff000);
00360 }
00361
00362
00363 /* test if APIC present */
00364 L4_INLINE int
00365 apic_test_present(void)
00366 {
00367 unsigned int dummy;
00368 unsigned int capability;
00369
00370 asm volatile("pushl %%ebx; cpuid; popl %%ebx"
00371 : "=a" (dummy),
00372 "=c" (dummy),
00373 "=d" (capability)
00374 : "a" (0x00000001)
00375 : "cc");
00376
00377 return ((capability & 1<<9) !=0);
00378 }
00379
00380
00381 L4_INLINE void
00382 apic_soft_enable(void)
00383 {
00384 unsigned long tmp_val;
00385 tmp_val = apic_read(APIC_SPIV);
00386 tmp_val |= (1<<8); /* enable APIC */
00387 tmp_val &= ~(1<<9); /* enable Focus Processor Checking */

```

```

00388 tmp_val |= 0xff; /* Set spurious IRQ vector to 0xff */
00389 apic_write(APIC_SPIV, tmp_val);
00390 }
00391
00392
00393 L4_INLINE void
00394 apic_init(unsigned long base_addr)
00395 {
00396 apic_map_base = base_addr;
00397 }
00398
00399
00400 L4_INLINE void
00401 apic_done(void)
00402 {
00403 apic_map_base = 0;
00404 }
00405
00406
00407 L4_INLINE void
00408 apic_irq_ack(void)
00409 {
00410 apic_read(APIC_SPIV);
00411 apic_write(APIC_EOI, 0);
00412 }
00413
00414
00415 #endif /* __L4_UTIL_APIC_H */

```

## 15.5 amd64/l4/util/idt.h File Reference

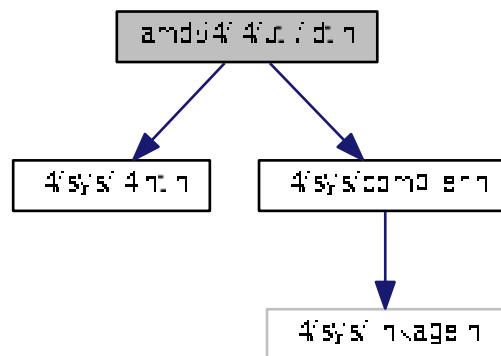
IDT related functions.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for idt.h:



## Data Structures

- struct [l4util\\_idt\\_desc\\_t](#)  
*IDT entry.*
- struct [l4util\\_idt\\_header\\_t](#)  
*Header of an IDT table.*

### 15.5.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).

## 15.6 idt.h

```

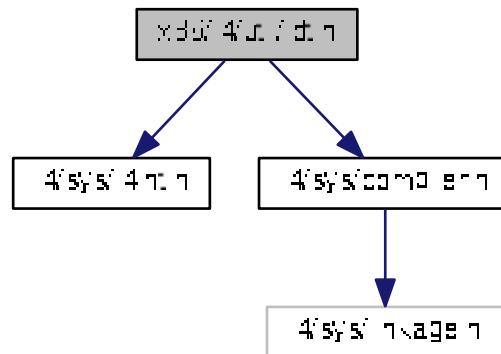
00001
00009 /*
00010 * (c) 2003-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00030
00033 typedef struct
00034 {
00035 l4_uint64_t a, b;
00036 } __attribute__((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042 l4_uint16_t limit;
00043 void *base;
00044 l4util_idt_desc_t desc[0];
00045 } __attribute__((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055 idt->desc[nr].a = (l4_uint64_t)handler & 0x0000ffff;
00056 idt->desc[nr].b = 0x0000ef00 | ((l4_uint64_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066 int i;
00067 idt->limit = entries*8 - 1;
00068 idt->base = &idt->desc;
00069
00070 for (i=0; i<entries; i++)
00071 l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080 asm volatile ("lidt (%rax) \n\t" : : "a" (idt));
00081 }
00083 EXTERN_C_END
00084
00085 #endif
00086

```

## 15.7 x86/l4/util/idt.h File Reference

IDT related functions.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
Include dependency graph for idt.h:
```



### Data Structures

- struct [l4util\\_idt\\_desc\\_t](#)  
*IDT entry.*
- struct [l4util\\_idt\\_header\\_t](#)  
*Header of an IDT table.*

### 15.7.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).

## 15.8 idt.h

```

00001
00009 /*
00010 * (c) 2003-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00030
00033 typedef struct
00034 {
00035 l4_uint32_t a, b;
00036 } __attribute__((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042 l4_uint16_t limit;
00043 void *base;
00044 l4util_idt_desc_t desc[0];
00045 } __attribute__((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055 idt->desc[nr].a = (l4_uint32_t)handler & 0x0000ffff;
00056 idt->desc[nr].b = 0x0000ef00 | ((l4_uint32_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066 int i;
00067 idt->limit = entries*8 - 1;
00068 idt->base = &idt->desc;
00069
00070 for (i=0; i<entries; i++)
00071 l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080 asm volatile ("lidt (%eax) \n\t" : : "a" (idt));
00081 }
00082
00084 EXTERN_C_END
00085
00086 #endif
00087

```

## 15.9 amd64/l4/util/perform.h File Reference

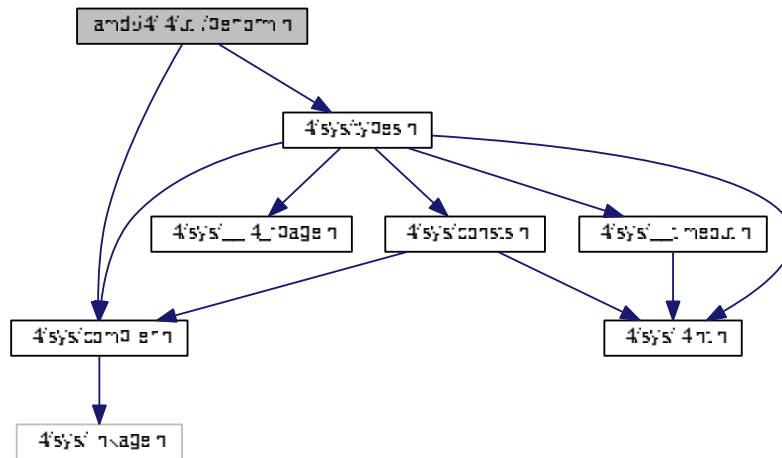
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for `perform.h`:



### 15.9.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 15.10 `perform.h`

```

00001
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015 #ifndef __L4UTIL_PERFORM_H
00016 #define __L4UTIL_PERFORM_H
00017
00018 #include <14/sys/types.h>
00019 #include <14/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 extern const char*strp6pmc_event(14_uint32_t event);
00024
00025 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00026
00027 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00028
00029 #error You must define your target architecture.
00030 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00031
00032 #else
00033
00034 /* P5/P6/K7 section */
00035
00036 /* Makros for access to model specific registers (MSR) */

```

```

00037
00038 /* Write the 64-Bit Model Specific Register. First argument is the register,
00039 second the 64-Bit value. This can only be called at privilege level 0.
00040 With L4, the kernel emulates the WRMSR when calling in PL 3.
00041 */
00042 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00043 unsigned long dummyeax, dummyecx, dummyedx;
00044
00045 asm volatile(
00046 ".byte 0xf; .byte 0x30\n" /* wrmsr */
00047 : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00048 : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00049);
00050 }
00051
00052 /* Read the 64-Bit Model Specific Register. First argument is the register,
00053 second the address to a 64-Bit value. This can only be called at
00054 privilege level 0. With L4, the kernel emulates the RDMSR when calling
00055 in PL 3.
00056 */
00057 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00058 unsigned dummy;
00059
00060 asm volatile(
00061 ".byte 0xf; .byte 0x32\n" /* rdmsr */
00062 : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00063 : "2" (reg)
00064);
00065 }
00066
00067
00068 #ifdef CPU_PENTIUM
00069 /* Pentium section */
00070
00071 /* functions and events defined here are only usable at Pentium
00072 Processors. P6 architecture does NOT support this kind of measuring and
00073 these events. P6 architecture has its own counters and its own events.
00074 See P6-section for details. */
00075
00076 /* from l4linux/arch/l4-i386/include/perform.h */
00077
00078 static inline void
00079 l4_i586_reset_event_counter(void){
00080 asm volatile("xor %%rax, %%rax\n"
00081 "xor %%rdx, %%rdx\n"
00082 "mov $0x12, %%rcx\n"
00083 ".byte 0x0f, 0x30\n"
00084 "movl $0x13, %%rcx\n"
00085 ".byte 0x0f, 0x30\n"
00086 : : "cx", "ax", "dx"
00087);
00088 };
00089
00090 static inline void
00091 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00092 {
00093 asm volatile(
00094 /* "movl $0, %%eax\n"
00095 "movl $0x11, %%ecx\n"
00096 ".byte 0x0f, 0x30\n" *//* stop event counting */
00097 "mov $0x12, %%rcx\n"
00098 ".byte 0x0f, 0x32\n"
00099 "mov %%rax, (%%%rbx)\n"
00100 "mov %%rdx, 4(%%rbx)\n"
00101 "mov $0x13, %%ecx\n"
00102 ".byte 0x0f, 0x32\n"
00103 "mov %%rax, (%%rsi)\n"
00104 "mov %%rdx, 4(%%rsi)\n"
00105 : /* no output */
00106 : "b" (counter0), "S" (counter1)
00107 : "ax", "cx", "dx"
00108);
00109 }
00110
00111 static inline void
00112 l4_i586_read_event_counter(int *counter0, int *counter1)
00113 {
00114 asm volatile("push %%rdx\n"
00115 ".byte 0x0f, 0x30\n"
00116 "mov $0x12, %%rcx\n"
00117 ".byte 0x0f, 0x32\n"
00118 "mov %%rax, %%rbx\n"
00119 "movl $0x13, %%rcx\n"
00120 ".byte 0x0f, 0x32\n"
00121 "popl %%edx\n"
00122 : "=b" (*counter0), "=a" (*counter1)
00123 : "1" (0), "c" (0x11)

```

```

00124);
00125 }
00126
00127 static inline void
00128 l4_i586_select_event(int event0, int event1)
00129 {
00130 asm volatile(".byte 0x0f, 0x30\n"
00131 :
00132 :
00133 "a" (event0 + (event1 << 16)),
00134 "d" (0),
00135 "c" (0x11)
00136);
00137 };
00138
00139 #define P5_RD_MISS 0x003 /* 000011B */
00140 #define P5_WR_MISS 0x008 /* 000100B */
00141 #define P5_RW_MISS 0x029 /* 101001B */
00142 #define P5_EX_MISS 0x00e /* 001110B */
00143
00144 #define P5_D_WBACK 0x006 /* 000110B */
00145
00146 #define P5_RW_TLB 0x002 /* 00010B */
00147 #define P5_EX_TLB 0x00d /* 01101B */
00148
00149 #define P5_A_STALL 0x01f /* 11111B */
00150 #define P5_W_STALL 0x019 /* 11001B */
00151 #define P5_R_STALL 0x01a /* 11010B */
00152 #define P5_X_STALL 0x01b /* 11011B */
00153
00154 #define P5_AGI_STALL 0x01f /* 11111B */
00155
00156 #define P5_PIPELINE_FLUSH 0x015 /* 10101B */
00157
00158 #define P5_NON_CACHE_RD 0x01e /* 11110B */
00159 #define P5_NCACHE_REFS 0x01e /* 11110B */
00160 #define P5_LOCKED_BUS 0x01c /* 11100B */
00161
00162 #define P5_MEM2PIPE 0x009 /* 01001B */
00163 #define P5_BANK_CONF 0x00a /* 01010B */
00164
00165
00166 #define P5_INSTRS_EX 0x016 /* 10110B */
00167 #define P5_INSTRS_EX_V 0x017 /* 10111B */
00168
00169
00170 #define P5_CNT_NOTHING (0x00 << 6) /* 00B << 6 */
00171 #define P5_CNT_EVENT_PL0 (0x01 << 6) /* 01B << 6 */
00172 #define P5_CNT_EVENT_PL3 (0x02 << 6) /* 10B << 6 */
00173 #define P5_CNT_EVENT (0x03 << 6) /* 11B << 6 */
00174 #define P5_CNT_CLOCKS_PL0 (0x05 << 6) /* 101B << 6 */
00175 #define P5_CNT_CLOCKS_PL3 (0x06 << 6) /* 110B << 6 */
00176 #define P5_CNT_CLOCKS (0x07 << 6) /* 111B << 6 */
00177
00178
00179 #else
00180 #if defined CPU_P6
00181 /* PPro/PII/PIII section */
00182
00183 /*-
00184 * Copyright (c) 1997 The President and Fellows of Harvard College.
00185 * All rights reserved.
00186 * Copyright (c) 1997 Aaron B. Brown.
00187 *
00188 * Redistribution and use in source and binary forms, with or without
00189 * modification, are permitted provided that the following conditions
00190 * are met:
00191 * 1. Redistributions of source code must retain the above copyright
00192 * notice, this list of conditions and the following disclaimer.
00193 * 2. Redistributions in binary form must reproduce the above copyright
00194 * notice, this list of conditions and the following disclaimer in the
00195 * documentation and/or other materials provided with the distribution.
00196 * 3. All advertising materials mentioning features or use of this software
00197 * must display the following acknowledgement:
00198 * This product includes software developed by Harvard University
00199 * and its contributors.
00200 * 4. Neither the name of the University nor the names of its contributors
00201 * may be used to endorse or promote products derived from this software
00202 * without specific prior written permission.
00203 *
00204 * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS'' AND
00205 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00206 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00207 * ARE DISCLAIMED. IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00208 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00209 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00210 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

```



```

00211 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00212 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00213 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00214 * POSSIBILITY OF SUCH DAMAGE.
00215 */
00216
00217 /*****
00218 ** Symbolic names for counter numbers (used in select_p6counter()) **
00219 *****/
00220 *
00221 * These correspond in order to the Pentium Pro counters. Add new counters at
00222 * the end. These agree with the mnemonics in the Pentium Pro Family
00223 * Developer's Manual, vol 3.
00224 *
00225 * Those events marked with a $ require a MESI unit field; those marked with
00226 * a @ require a self/any unit field. Those marked with a 0 are only supported
00227 * in counter 0; those marked with 1 are only supported in counter 1.
00228 */
00229
00230 /* Data cache unit */
00231 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00232 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00233 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00234 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00235 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00236
00237 /* Instruction fetch unit */
00238 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00239 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00240 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00241 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00242 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00243
00244 /* L2 Cache */
00245 #define P6_L2_IFETCH 0x28 /* ($) 12 ifetches */
00246 #define P6_L2_LD 0x29 /* ($) 12 data loads */
00247 #define P6_L2_ST 0x2a /* ($) 12 data stores */
00248 #define P6_L2_LINES_IN 0x24 /* lines allocated in L2 */
00249 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00250 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00251 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00252 #define P6_L2_RQSTS 0x2e /* ($) number of 12 requests */
00253 #define P6_L2_ADS 0x21 /* number of 12 addr strobes */
00254 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00255 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring 12->cpu */
00256
00257 /* External bus logic */
00258 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00259 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00260 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00261 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00262 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00263 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */
00264 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00265 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00266 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00267 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00268 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00269 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00270 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00271 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00272 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00273 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00274 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00275 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00276 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00277 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00278
00279 /* FPU */
00280 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00281 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00282 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00283 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00284 #define P6_DIV 0x13 /* (1) number of FP divides */
00285 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00286
00287 /* Memory ordering */
00288 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00289 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00290 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00291
00292 /* Instruction decoding and retirement */
00293 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00294 #define P6_UOPS_RETIRED 0xc2 /* number of micro-ops retired */
00295 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */
00296
00297 /* Interrupts */

```



```

00385
00386 static inline l4_uint32_t l4_i686_rdpmc_32(int cntnr){
00387 l4_uint32_t x;
00388
00389 __asm__ __volatile__(
00390 ".byte 0xf; .byte 0x33 # RDPMC instruction"
00391 : "=a" (x)
00392 : "c" (cntnr)
00393 : "rcx", "rax", "rdx");
00394 return x;
00395 }
00396
00397 static inline void l4_i686_select_perfctr_event(int counter,
00398 unsigned long long val){
00399 l4_i586_wrmsr(MSR_P6_EVTSEL0+counter, &val);
00400 }
00401
00402 static inline void l4_i686_select_perfctr0_event(long long *val){
00403 __asm volatile(
00404 "mov $MSR_P6_EVTSEL0, %%rcx\n"
00405 "mov (%%rbx), %%rax\n"
00406 "mov 4(%%rbx), %%rdx\n"
00407 /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00408 ".byte 0x0f, 0x30\n" // wrmsr
00409 /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00410 : /* no output */
00411 : "b" (val)
00412 : "ax", "cx", "dx", "bx"
00413);
00414
00415 }
00416
00417 /* end of P6 section */
00418 #else
00419
00420 #define K7CNT_U 0x010000 /* Monitor user-level events */
00421 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00422 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00423 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00424 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00425 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00426 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00427 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00428
00429 #define MSR_K7_EVTSEL0 0xC0010000
00430 #define MSR_K7_EVTSEL1 0xC0010001
00431 #define MSR_K7_EVTSEL2 0xC0010002
00432 #define MSR_K7_EVTSEL3 0xC0010003
00433 #define MSR_K7_PERFCTR0 0xC0010004
00434 #define MSR_K7_PERFCTR1 0xC0010005
00435 #define MSR_K7_PERFCTR2 0xC0010006
00436 #define MSR_K7_PERFCTR3 0xC0010007
00437
00438 #endif
00439
00440 #endif
00441
00442 /* end of P5/P6/K7 section*/
00443 #endif
00444
00445 /* end of not only lib-prototypes section */
00446 #endif
00447
00448 EXTERN_C_END
00449
00450 #endif

```

## 15.11 x86/I4/util/perform.h File Reference

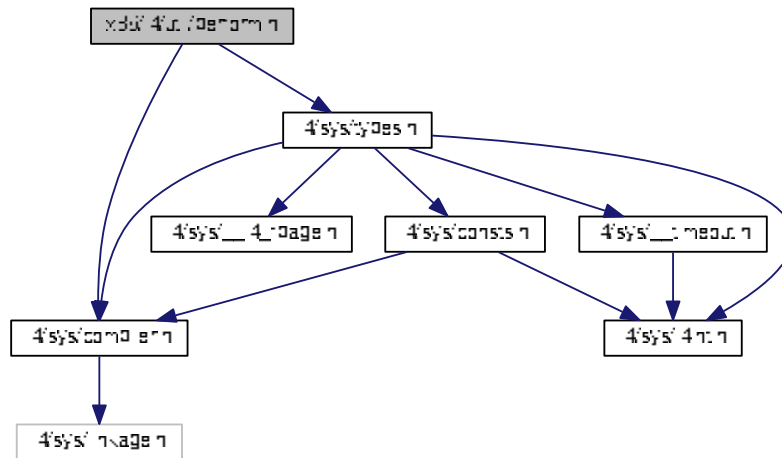
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for `perform.h`:



### 15.11.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 15.12 `perform.h`

```

00001
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010 * Lars Reuther <reuther@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL_PERFORM_H
00018 #define __L4UTIL_PERFORM_H
00019
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 extern const char*strp6pmc_event(l4_uint32_t event);
00026
00027 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00028
00029 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00030
00031 #error You must define your target architecture.
00032 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00033
00034 #else
00035
00036 /* P5/P6/K7 section */

```

```

00037
00038 /* Makros for access to model specific registers (MSR) */
00039
00040 /* Write the 64-Bit Model Specific Register. First argument is the register,
00041 second the 64-Bit value. This can only be called at privilege level 0.
00042 With L4, the kernel emulates the WRMSR when calling in PL 3.
00043 */
00044 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00045 unsigned long dummyeax, dummyecx, dummyedx;
00046
00047 asm volatile(
00048 ".byte 0xf; .byte 0x30\n" /* wrmsr */
00049 : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00050 : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00051);
00052 }
00053
00054 /* Read the 64-Bit Model Specific Register. First argument is the register,
00055 second the address to a 64-Bit value. This can only be called at
00056 privilege level 0. With L4, the kernel emulates the RDMSR when calling
00057 in PL 3.
00058 */
00059 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00060 unsigned dummy;
00061
00062 asm volatile(
00063 ".byte 0xf; .byte 0x32\n" /* rdmsr */
00064 : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00065 : "2" (reg)
00066);
00067 }
00068
00069
00070 #ifdef CPU_PENTIUM
00071 /* Pentium section */
00072
00073 /* functions and events defined here are only usable at Pentium
00074 Processors. P6 architecture does NOT support this kind of measuring and
00075 these events. P6 architecture has its own counters and its own events.
00076 See P6-section for details. */
00077
00078 /* from l4linux/arch/l4-i386/include/perform.h */
00079
00080 static inline void
00081 l4_i586_reset_event_counter(void){
00082 asm volatile("xor %%eax, %%eax\n"
00083 "xor %%edx, %%edx\n"
00084 "movl $0x12, %%ecx\n"
00085 ".byte 0x0f, 0x30\n"
00086 "movl $0x13, %%ecx\n"
00087 ".byte 0x0f, 0x30\n"
00088 : : : "cx", "ax", "dx"
00089);
00090 };
00091
00092 static inline void
00093 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00094 {
00095 asm volatile(
00096 /* "movl $0, %%eax\n"
00097 "movl $0x11, %%ecx\n"
00098 ".byte 0x0f, 0x30\n" */ /* stop event counting */
00099 "movl $0x12, %%ecx\n"
00100 ".byte 0x0f, 0x32\n"
00101 "movl %%eax, (%ebx)\n"
00102 "movl %%edx, 4(%ebx)\n"
00103 "movl $0x13, %%ecx\n"
00104 ".byte 0x0f, 0x32\n"
00105 "movl %%eax, (%esi)\n"
00106 "movl %%edx, 4(%esi)\n"
00107 : /* no output */
00108 : "b" (counter0), "S" (counter1)
00109 : "ax", "cx", "dx"
00110);
00111 }
00112
00113 static inline void
00114 l4_i586_read_event_counter(int *counter0, int *counter1)
00115 {
00116 asm volatile("pushl %%edx\n"
00117 ".byte 0x0f, 0x30\n"
00118 "movl $0x12, %%ecx\n"
00119 ".byte 0x0f, 0x32\n"
00120 "movl %%eax, %%ebx\n"
00121 "movl $0x13, %%ecx\n"
00122 ".byte 0x0f, 0x32\n"
00123 "popl %%edx\n"

```

```

00124 : "b" (*counter0), "a" (*counter1)
00125 : "1" (0), "c" (0x11)
00126);
00127 }
00128
00129 static inline void
00130 l4_i586_select_event(int event0, int event1)
00131 {
00132 asm volatile(".byte 0x0f, 0x30\n"
00133 :
00134 :
00135 "a" (event0 + (event1 << 16)),
00136 "d" (0),
00137 "c" (0x11)
00138);
00139 };
00140
00141 #define P5_RD_MISS 0x003 /* 000011B */
00142 #define P5_WR_MISS 0x008 /* 000100B */
00143 #define P5_RW_MISS 0x029 /* 101001B */
00144 #define P5_EX_MISS 0x00e /* 001110B */
00145
00146 #define P5_D_WBACK 0x006 /* 000110B */
00147
00148 #define P5_RW_TLB 0x002 /* 00010B */
00149 #define P5_EX_TLB 0x00d /* 01101B */
00150
00151 #define P5_A_STALL 0x01f /* 11111B */
00152 #define P5_W_STALL 0x019 /* 11001B */
00153 #define P5_R_STALL 0x01a /* 11010B */
00154 #define P5_X_STALL 0x01b /* 11011B */
00155
00156 #define P5_AGI_STALL 0x01f /* 11111B */
00157
00158 #define P5_PIPELINE_FLUSH 0x015 /* 10101B */
00159
00160 #define P5_NON_CACHE_RD 0x01e /* 11110B */
00161 #define P5_NCACHE_REFS 0x01e /* 11110B */
00162 #define P5_LOCKED_BUS 0x01c /* 11100B */
00163
00164 #define P5_MEM2PIPE 0x009 /* 01001B */
00165 #define P5_BANK_CONF 0x00a /* 01010B */
00166
00167
00168 #define P5_INSTRS_EX 0x016 /* 10110B */
00169 #define P5_INSTRS_EX_V 0x017 /* 10111B */
00170
00171
00172 #define P5_CNT_NOTHING (0x00 << 6) /* 00B << 6 */
00173 #define P5_CNT_EVENT_PL0 (0x01 << 6) /* 01B << 6 */
00174 #define P5_CNT_EVENT_PL3 (0x02 << 6) /* 10B << 6 */
00175 #define P5_CNT_EVENT (0x03 << 6) /* 11B << 6 */
00176 #define P5_CNT_CLOCKS_PL0 (0x05 << 6) /* 101B << 6 */
00177 #define P5_CNT_CLOCKS_PL3 (0x06 << 6) /* 110B << 6 */
00178 #define P5_CNT_CLOCKS (0x07 << 6) /* 111B << 6 */
00179
00180
00181 #else
00182 #if defined CPU_P6
00183 /* PPro/PII/PIII section */
00184
00185 /*-
00186 * Copyright (c) 1997 The President and Fellows of Harvard College.
00187 * All rights reserved.
00188 * Copyright (c) 1997 Aaron B. Brown.
00189 *
00190 * Redistribution and use in source and binary forms, with or without
00191 * modification, are permitted provided that the following conditions
00192 * are met:
00193 * 1. Redistributions of source code must retain the above copyright
00194 * notice, this list of conditions and the following disclaimer.
00195 * 2. Redistributions in binary form must reproduce the above copyright
00196 * notice, this list of conditions and the following disclaimer in the
00197 * documentation and/or other materials provided with the distribution.
00198 * 3. All advertising materials mentioning features or use of this software
00199 * must display the following acknowledgement:
00200 * This product includes software developed by Harvard University
00201 * and its contributors.
00202 * 4. Neither the name of the University nor the names of its contributors
00203 * may be used to endorse or promote products derived from this software
00204 * without specific prior written permission.
00205 *
00206 * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS'' AND
00207 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00208 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00209 * ARE DISCLAIMED. IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00210 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

```

```

00211 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00212 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00213 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00214 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00215 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00216 * POSSIBILITY OF SUCH DAMAGE.
00217 */
00218
00219 /*****
00220 ** Symbolic names for counter numbers (used in select_p6counter()) **
00221 *****/
00222 *
00223 * These correspond in order to the Pentium Pro counters. Add new counters at
00224 * the end. These agree with the mnemonics in the Pentium Pro Family
00225 * Developer's Manual, vol 3.
00226 *
00227 * Those events marked with a $ require a MESI unit field; those marked with
00228 * a @ require a self/any unit field. Those marked with a 0 are only supported
00229 * in counter 0; those marked with 1 are only supported in counter 1.
00230 */
00231
00232 /* Data cache unit */
00233 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00234 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00235 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00236 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00237 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00238
00239 /* Instruction fetch unit */
00240 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00241 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00242 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00243 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00244 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00245
00246 /* L2 Cache */
00247 #define P6_L2_IFETCH 0x28 /* ($) L2 ifetches */
00248 #define P6_L2_LD 0x29 /* ($) L2 data loads */
00249 #define P6_L2_ST 0x2a /* ($) L2 data stores */
00250 #define P6_L2_LINES_IN 0x24 /* lines allocated in L2 */
00251 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00252 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00253 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00254 #define P6_L2_RQSTS 0x2e /* ($) number of L2 requests */
00255 #define P6_L2_ADS 0x21 /* number of L2 addr strobes */
00256 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00257 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring L2->CPU */
00258
00259 /* External bus logic */
00260 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00261 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00262 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00263 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00264 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00265 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */
00266 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00267 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00268 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00269 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00270 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00271 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00272 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00273 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00274 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00275 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00276 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00277 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00278 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00279 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00280
00281 /* FPU */
00282 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00283 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00284 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00285 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00286 #define P6_DIV 0x13 /* (1) number of FP divides */
00287 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00288
00289 /* Memory ordering */
00290 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00291 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00292 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00293
00294 /* Instruction decoding and retirement */
00295 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00296 #define P6_UOPS_RETIRED 0xc2 /* number of micro-ops retired */
00297 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */

```

```

00298
00299 /* Interrupts */
00300 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */
00301 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00302 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00303
00304 /* Branches */
00305 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00306 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00307 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00308 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired*/
00309 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00310 #define P6_BTБ_MISSES 0xe2 /* # of branches that missed in BTБ */
00311 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00312 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00313
00314 /* Stalls */
00315 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00316 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls*/
00317
00318 /* Segment register loads */
00319 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00320
00321 /* Clocks */
00322 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00323
00324 /* Unit field tags */
00325 #define P6_UNIT_M 0x0800
00326 #define P6_UNIT_E 0x0400
00327 #define P6_UNIT_S 0x0200
00328 #define P6_UNIT_I 0x0100
00329 #define P6_UNIT_MESI 0x0f00
00330
00331 #define P6_UNIT_SELF 0x0000
00332 #define P6_UNIT_ANY 0x2000
00333
00334 /*****
00335 ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00336 *****/
00337 *
00338 * The driver accepts fully-formed counter specifications from user-level.
00339 * The following flags are mnemonics for the bits that get set in the
00340 * PerfEvtSel0 and PerfEvtSel1 MSR's
00341 *
00342 */
00343 #define P6CNT_U 0x010000 /* Monitor user-level events */
00344 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00345 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00346 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00347 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00348 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00349 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00350 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */
00351
00352 /*****
00353 ** Miscellaneous constants **
00354 *****/
00355 *
00356 * Number of Pentium Pro programable hardware counters.
00357 */
00358 #define NUM_P6HWC 2
00359
00360 /*****
00361 *
00362 * End of Copyright by Harvard College
00363 *
00364 *****/
00365
00366
00367 #define MSR_P6_EVTSEL0 0x186
00368 #define MSR_P6_EVTSEL1 0x187
00369 #define MSR_P6_PERFCTR0 0xc1
00370 #define MSR_P6_PERFCTR1 0xc2
00371
00372 /* P6-specific Makros to manipulate and read counters */
00373
00374 /* Read the 40 bit performance monitoring counter. This requires
00375 the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
00376 at P6.
00377 */
00378 #define l4_i686_rdpmc(cnr, res_p) \
00379 __asm __volatile(\
00380 "movl %2, %%ecx # put counter number in \n\
00381 .byte 0xf; .byte 0x33 # RDPMC instruction \n\
00382 movl %%edx, %1 # High order 32 bits \n\
00383 movl %%eax, %0 # Low order 32 bits" \
00384 : "=g" (*(int *) (res_p)), "=g" (((int *) res_p)+1)) \

```



```

00385 : "g" (cnt) \
00386 : "ecx", "eax", "edx")
00387
00388 static inline l4_uint32_t l4_i686_rdtsc_32(int cnt){
00389 l4_uint32_t x;
00390
00391 __asm__ __volatile__(
00392 ".byte 0xf; .byte 0x33 # RDTSC instruction"
00393 : "=a" (x)
00394 : "c" (cnt)
00395 : "ecx", "eax", "edx");
00396 return x;
00397 }
00398
00399 static inline void l4_i686_select_perfctr_event(int counter,
00400 unsigned long long val){
00401 l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
00402 }
00403
00404 static inline void l4_i686_select_perfctr0_event(long long *val){
00405 asm volatile(
00406 "movl $MSR_P6_EVNTSEL0, %%ecx\n"
00407 "movl (%ebx), %%eax\n"
00408 "movl 4(%%ebx), %%edx\n"
00409 /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00410 ".byte 0x0f, 0x30\n" // wrmsr
00411 /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00412 : /* no output */
00413 : "b" (val)
00414 : "ax", "cx", "dx", "bx"
00415);
00416
00417 }
00418
00419 /* end of P6 section */
00420 #else
00421
00422 #define K7CNT_U 0x010000 /* Monitor user-level events */
00423 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00424 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00425 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00426 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00427 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00428 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00429 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00430
00431 #define MSR_K7_EVNTSEL0 0xC0010000
00432 #define MSR_K7_EVNTSEL1 0xC0010001
00433 #define MSR_K7_EVNTSEL2 0xC0010002
00434 #define MSR_K7_EVNTSEL3 0xC0010003
00435 #define MSR_K7_PERFCTR0 0xC0010004
00436 #define MSR_K7_PERFCTR1 0xC0010005
00437 #define MSR_K7_PERFCTR2 0xC0010006
00438 #define MSR_K7_PERFCTR3 0xC0010007
00439
00440 #endif
00441
00442 #endif
00443
00444 /* end of P5/P6/K7 section */
00445 #endif
00446
00447 /* end of not only lib-prototypes section */
00448 #endif
00449
00450 EXTERN_C_END
00451
00452 #endif

```

## 15.13 amd64/l4/util/rdtsc.h File Reference

time stamp counter related functions

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```



- void [l4\\_busy\\_wait\\_ns](#) (l4\_uint64\_t ns)  
*Wait busy for a small amount of time.*
- void [l4\\_busy\\_wait\\_us](#) (l4\_uint64\_t us)  
*Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc](#) (l4\_kernel\_info\_t \*kip)  
*Calibrate scalers for time stamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init](#) (int constraint, l4\_kernel\_info\_t \*kip)  
*Initialitze scaler for TSC calicaltions.*
- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

### 15.13.1 Detailed Description

time stamp counter related functions

Date

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 15.14 rdtsc.h

```

00001
00009 /*
00010 * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00012 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00013 * economic rights: Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 #ifndef __l4_rdtsc_h
00020 #define __l4_rdtsc_h
00021
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/kip.h>
00030
00031 EXTERN_C_BEGIN
00032
00033 /* interface */
00038
00039 #define L4_TSC_INIT_AUTO 0
00040 #define L4_TSC_INIT_KERNEL 1
00041 #define L4_TSC_INIT_CALIBRATE 2
00042
00043 extern l4_uint32_t l4_scaler_tsc_to_ns;
00044 extern l4_uint32_t l4_scaler_tsc_to_us;
00045 extern l4_uint32_t l4_scaler_ns_to_tsc;
00046 extern l4_uint32_t l4_scaler_tsc_linux;
00047
00052 L4_INLINE l4_cpu_time_t
00053 l4_rdtsc (void);
00054
00060 L4_INLINE
00061 l4_uint32_t l4_rdtsc_32 (void);
00062
00067 L4_INLINE l4_cpu_time_t
00068 l4_rdpmc (int nr);
00069
00075 L4_INLINE
00076 l4_uint32_t l4_rdpmc_32 (int nr);
00077

```

```

00082 L4_INLINE l4_uint64_t
00083 l4_tsc_to_ns (l4_cpu_time_t tsc);
00084
00089 L4_INLINE l4_uint64_t
00090 l4_tsc_to_us (l4_cpu_time_t tsc);
00091
00097 L4_INLINE void
00098 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00099 l4_uint32_t *s, l4_uint32_t *ns);
00099
00105 L4_INLINE l4_cpu_time_t
00106 l4_ns_to_tsc (l4_uint64_t ns);
00107
00113 L4_INLINE void
00114 l4_busy_wait_ns (l4_uint64_t ns);
00115
00121 L4_INLINE void
00122 l4_busy_wait_us (l4_uint64_t us);
00123
00124 EXTERN_C_BEGIN
00125
00134 L4_INLINE l4_uint32_t
00135 l4_calibrate_tsc (l4_kernel_info_t *kip);
00136
00162 L4_CV l4_uint32_t
00163 l4_tsc_init (int constraint, l4_kernel_info_t *kip);
00164
00169 L4_CV l4_uint32_t
00170 l4_get_hz (void);
00171
00174 EXTERN_C_END
00175
00176 /* implementaion */
00177
00178 L4_INLINE l4_uint32_t
00179 l4_calibrate_tsc (l4_kernel_info_t *kip)
00180 {
00181 return l4_tsc_init(L4_TSC_INIT_AUTO, kip);
00182 }
00183
00184 L4_INLINE l4_cpu_time_t
00185 l4_rdtsc (void)
00186 {
00187 l4_cpu_time_t v;
00188
00189 __asm__ __volatile__ (
00190 ("byte 0x0f, 0x31\n\t"
00191 "mov $0xffffffff, %%rcx\n\t"
00192 "and %%rcx, %%rax\n\t"
00193 "shlq $32, %%rdx\n\t"
00194 "orq %%rdx, %%rax\n\t"
00195 :
00196 "=a" (v)
00197 : /* no inputs */
00198 : "rdx", "rcx"
00199);
00200
00201 return v;
00202 }
00203
00204 L4_INLINE l4_cpu_time_t
00205 l4_rdpmc (int nr)
00206 {
00207 l4_cpu_time_t v;
00208 l4_uint64_t dummy;
00209
00210 __asm__ __volatile__ (
00211 "rdpmc\n\t"
00212 "mov $0xffffffff, %%rcx\n\t"
00213 "and %%rcx, %%rax\n\t"
00214 "shlq $32, %%rdx\n\t"
00215 "orq %%rdx, %%rax\n\t"
00216 :
00217 "=a" (v), "=c" (dummy)
00218 : "c" (nr)
00219 : "rdx"
00220);
00221
00222 return v;
00223 }
00224
00225 /* the same, but only 32 bit. Useful for smaller differences */
00226 L4_INLINE
00227 l4_uint32_t l4_rdpmc_32(int nr)
00228 {
00229 l4_uint32_t x;
00230 l4_uint64_t dummy;

```

```

00231
00232 __asm__ __volatile__ (
00233 "rdpmc \n\t"
00234 "mov $0xffffffff, %%rcx \n\t" /* clears the upper 32 bits! */
00235 "and %%rcx, %%rax \n\t"
00236 : "=a" (x), "=c" (dummy)
00237 : "c" (nr)
00238 : "rdx");
00239
00240 return x;
00241 }
00242
00243 /* the same, but only 32 bit. Useful for smaller differences,
00244 needs less cycles. */
00245 L4_INLINE
00246 l4_uint32_t l4_rdtsc_32(void)
00247 {
00248 l4_uint32_t x;
00249
00250 __asm__ __volatile__ (
00251 ".byte 0x0f, 0x31\n\t" // rdtsc
00252 : "=a" (x)
00253 :
00254 : "rdx");
00255
00256 return x;
00257 }
00258
00259 L4_INLINE l4_uint64_t
00260 l4_tsc_to_ns (l4_cpu_time_t tsc)
00261 {
00262 l4_uint64_t ns, dummy;
00263
00264 __asm__
00265 (" \n\t"
00266 "mulq %3 \n\t"
00267 "shrd $27, %%rdx, %%rax \n\t"
00268 : "=a" (ns), "=d" (dummy)
00269 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns)
00270);
00271 return ns;
00272 }
00273 L4_INLINE l4_uint64_t
00274 l4_tsc_to_us (l4_cpu_time_t tsc)
00275 {
00276 l4_uint64_t ns, dummy;
00277
00278 __asm__
00279 (" \n\t"
00280 "mulq %3 \n\t"
00281 "shrd $32, %%rdx, %%rax \n\t"
00282 : "=a" (ns), "=d" (dummy)
00283 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_us)
00284);
00285 return ns;
00286 }
00287 L4_INLINE void
00288 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00289 l4_uint32_t *s, l4_uint32_t *ns)
00290 {
00291 __asm__
00292 (" \n\t"
00293 "mulq %3 \n\t"
00294 "shrd $27, %%rdx, %%rax \n\t"
00295 "xorq %%rdx, %%rdx \n\t"
00296 "divq %4 \n\t"
00297 : "=a" (*s), "=d" (*ns)
00298 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns),
00299 "rm" (1000000000ULL)
00300);
00301 }
00302 L4_INLINE l4_cpu_time_t
00303 l4_ns_to_tsc (l4_uint64_t ns)
00304 {
00305 l4_uint64_t tsc, dummy;
00306
00307 __asm__
00308 (" \n\t"
00309 "mulq %3 \n\t"
00310 "shrd $27, %%rdx, %%rax \n\t"
00311 : "=a" (tsc), "=d" (dummy)
00312 : "a" (ns), "r" ((l4_uint64_t)l4_scaler_ns_to_tsc)
00313);
00314 return tsc;
00315 }
00316 L4_INLINE void

```

```

00317 l4_busy_wait_ns (l4_uint64_t ns)
00318 {
00319 l4_cpu_time_t stop = l4_rdtsc();
00320 stop += l4_ns_to_tsc(ns);
00321
00322 while (l4_rdtsc() < stop)
00323 ;
00324 }
00325
00326 L4_INLINE void
00327 l4_busy_wait_us (l4_uint64_t us)
00328 {
00329 l4_cpu_time_t stop = l4_rdtsc ();
00330 stop += l4_ns_to_tsc(us*1000ULL);
00331
00332 while (l4_rdtsc() < stop)
00333 ;
00334 }
00335
00336 EXTERN_C_END
00337
00338 #endif /* __l4_rdtsc_h */
00339

```

## 15.15 x86/l4/util/rdtsc.h File Reference

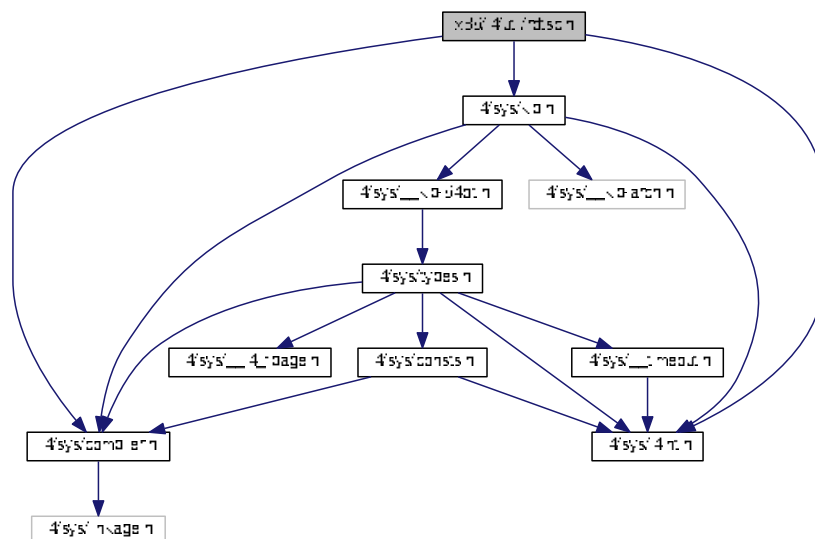
time stamp counter related functions

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```

Include dependency graph for rdtsc.h:



### Macros

- `#define L4_TSC_INIT_AUTO 0`  
*Automatic init.*
- `#define L4_TSC_INIT_KERNEL 1`  
*Initialized by kernel.*
- `#define L4_TSC_INIT_CALIBRATE 2`  
*Initialized by user-level.*

## Functions

- [l4\\_cpu\\_time\\_t l4\\_rdtsc](#) (void)  
*Read current value of CPU-internal time stamp counter.*
- [l4\\_uint32\\_t l4\\_rdtsc\\_32](#) (void)  
*Read the least significant 32 bit of the TSC.*
- [l4\\_cpu\\_time\\_t l4\\_rdpmc](#) (int nr)  
*Return current value of CPU-internal performance measurement counter.*
- [l4\\_uint32\\_t l4\\_rdpmc\\_32](#) (int nr)  
*Return the least significant 32 bit of a performance counter.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert time stamp to ns value.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_us](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert time stamp into micro seconds value.*
- void [l4\\_tsc\\_to\\_s\\_and\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc, [l4\\_uint32\\_t](#) \*s, [l4\\_uint32\\_t](#) \*ns)  
*Convert time stamp to s.ns value.*
- [l4\\_cpu\\_time\\_t l4\\_ns\\_to\\_tsc](#) ([l4\\_uint64\\_t](#) ns)  
*Convert nano seconds into CPU ticks.*
- void [l4\\_busy\\_wait\\_ns](#) ([l4\\_uint64\\_t](#) ns)  
*Wait busy for a small amount of time.*
- void [l4\\_busy\\_wait\\_us](#) ([l4\\_uint64\\_t](#) us)  
*Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc](#) ([l4\\_kernel\\_info\\_t](#) \*kip)  
*Calibrate scalers for time stamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init](#) (int constraint, [l4\\_kernel\\_info\\_t](#) \*kip)  
*Initialitze scaler for TSC calications.*
- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

### 15.15.1 Detailed Description

time stamp counter related functions

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 15.16 rdtsc.h

```

00001
00009 /*
00010 * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00012 * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00013 * Jork Löser <jork@os.inf.tu-dresden.de>,
00014 * Martin Pohlack <mp26@os.inf.tu-dresden.de>
00015 * economic rights: Technische Universität Dresden (Germany)
00016 * This file is part of TUD:OS and distributed under the terms of the
00017 * GNU Lesser General Public License 2.1.
00018 * Please see the COPYING-LGPL-2.1 file for details.
00019 */

```

```

00020
00021 #ifndef __l4_rdtsc_h
00022 #define __l4_rdtsc_h
00023
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031 #include <l4/sys/kip.h>
00032
00033 EXTERN_C_BEGIN
00034
00035 /* interface */
00040
00041 #define L4_TSC_INIT_AUTO 0
00042 #define L4_TSC_INIT_KERNEL 1
00043 #define L4_TSC_INIT_CALIBRATE 2
00044
00045 extern l4_uint32_t l4_scaler_tsc_to_ns;
00046 extern l4_uint32_t l4_scaler_tsc_to_us;
00047 extern l4_uint32_t l4_scaler_ns_to_tsc;
00048 extern l4_uint32_t l4_scaler_tsc_linux;
00049
00054 L4_INLINE l4_cpu_time_t
00055 l4_rdtsc (void);
00056
00062 L4_INLINE
00063 l4_uint32_t l4_rdtsc_32(void);
00064
00069 L4_INLINE l4_cpu_time_t
00070 l4_rdpmc (int nr);
00071
00077 L4_INLINE
00078 l4_uint32_t l4_rdpmc_32(int nr);
00079
00084 L4_INLINE l4_uint64_t
00085 l4_tsc_to_ns (l4_cpu_time_t tsc);
00086
00091 L4_INLINE l4_uint64_t
00092 l4_tsc_to_us (l4_cpu_time_t tsc);
00093
00099 L4_INLINE void
00100 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00101 l4_uint32_t *s, l4_uint32_t *ns);
00101
00107 L4_INLINE l4_cpu_time_t
00108 l4_ns_to_tsc (l4_uint64_t ns);
00109
00115 L4_INLINE void
00116 l4_busy_wait_ns (l4_uint64_t ns);
00117
00123 L4_INLINE void
00124 l4_busy_wait_us (l4_uint64_t us);
00125
00126 EXTERN_C_BEGIN
00127
00136 L4_INLINE l4_uint32_t
00137 l4_calibrate_tsc (l4_kernel_info_t *kip);
00138
00164 L4_CV l4_uint32_t
00165 l4_tsc_init (int constraint, l4_kernel_info_t *kip);
00166
00171 L4_CV l4_uint32_t
00172 l4_get_hz (void);
00173
00176 EXTERN_C_END
00177
00178 /* implementaion */
00179
00180 L4_INLINE l4_uint32_t
00181 l4_calibrate_tsc (l4_kernel_info_t *kip)
00182 {
00183 return l4_tsc_init(L4_TSC_INIT_AUTO, kip);
00184 }
00185
00186 L4_INLINE l4_cpu_time_t
00187 l4_rdtsc (void)
00188 {
00189 l4_cpu_time_t v;
00190
00191 __asm__ __volatile__
00192 (
00193 " .byte 0x0f, 0x31 \n\t"
00194 /*"rdtsc\n\t"*/
00195 :
00196 "=A" (v)
00197 : /* no inputs */
00198);
00199

```



```

00200 return v;
00201 }
00202
00203 /* the same, but only 32 bit. Useful for smaller differences,
00204 needs less cycles. */
00205 L4_INLINE
00206 l4_uint32_t l4_rdtsc_32(void)
00207 {
00208 l4_uint32_t x;
00209
00210 __asm__ __volatile__ (
00211 ".byte 0x0f, 0x31\n\t" // rdtsc
00212 : "=a" (x)
00213 :
00214 : "edx");
00215
00216 return x;
00217 }
00218
00219 L4_INLINE l4_cpu_time_t
00220 l4_rdpmc (int nr)
00221 {
00222 l4_cpu_time_t v;
00223
00224 __asm__ __volatile__ (
00225 "rdpmc\n\t"
00226 :
00227 : "=A" (v)
00228 : "c" (nr)
00229);
00230
00231 return v;
00232 }
00233
00234 /* the same, but only 32 bit. Useful for smaller differences,
00235 needs less cycles. */
00236 L4_INLINE
00237 l4_uint32_t l4_rdpmc_32(int nr)
00238 {
00239 l4_uint32_t x;
00240
00241 __asm__ __volatile__ (
00242 "rdpmc\n\t"
00243 : "=a" (x)
00244 : "c" (nr)
00245 : "edx");
00246
00247 return x;
00248 }
00249
00250 L4_INLINE l4_uint64_t
00251 l4_tsc_to_ns (l4_cpu_time_t tsc)
00252 {
00253 l4_uint32_t dummy;
00254 l4_uint64_t ns;
00255 __asm__
00256 (
00257 "\n\t"
00258 "movl %%edx, %%ecx\n\t"
00259 "mull %3\n\t"
00260 "movl %%ecx, %%eax\n\t"
00261 "movl %%edx, %%ecx\n\t"
00262 "mull %3\n\t"
00263 "addl %%ecx, %%eax\n\t"
00264 "adcl $0, %%edx\n\t"
00265 "shld $5, %%eax, %%edx\n\t"
00266 "shll $5, %%eax\n\t"
00267 : "=A" (ns),
00268 "=c" (dummy)
00269 : "0" (tsc),
00270 "g" (l4_scaler_tsc_to_ns)
00271);
00272 return ns;
00273 }
00274
00275 L4_INLINE l4_uint64_t
00276 l4_tsc_to_us (l4_cpu_time_t tsc)
00277 {
00278 l4_uint32_t dummy;
00279 l4_uint64_t us;
00280 __asm__
00281 (
00282 "\n\t"
00283 "movl %%edx, %%ecx\n\t"
00284 "mull %3\n\t"
00285 "movl %%ecx, %%eax\n\t"
00286 "movl %%edx, %%ecx\n\t"
00287 "mull %3\n\t"
00288 "addl %%ecx, %%eax\n\t"

```

```

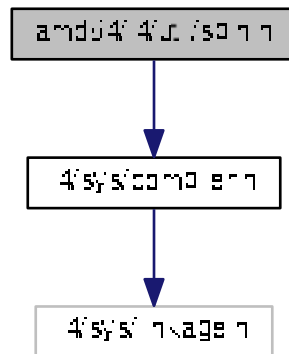
00287 "adcl $0, %%edx \n\t"
00288 : "=A" (us),
00289 "=&c" (dummy)
00290 : "0" (tsc),
00291 "g" (l4_scaler_tsc_to_us)
00292);
00293 return us;
00294 }
00295
00296 L4_INLINE void
00297 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00298 l4_uint32_t *s, l4_uint32_t *ns)
00299 {
00300 l4_uint32_t dummy;
00301 __asm__
00302 (
00303 "movl %%edx, %%ecx \n\t"
00304 "mull %4 \n\t"
00305 "movl %%ecx, %%eax \n\t"
00306 "movl %%edx, %%ecx \n\t"
00307 "mull %4 \n\t"
00308 "addl %%ecx, %%eax \n\t"
00309 "adcl $0, %%edx \n\t"
00310 "movl $1000000000, %%ecx \n\t"
00311 "shld $5, %%eax, %%edx \n\t"
00312 "shll $5, %%eax \n\t"
00313 "divl %%ecx \n\t"
00314 : "=a" (*s), "=d" (*ns), "=&c" (dummy)
00315 : "A" (tsc), "g" (l4_scaler_tsc_to_ns)
00316);
00317 }
00318 L4_INLINE l4_cpu_time_t
00319 l4_ns_to_tsc (l4_uint64_t ns)
00320 {
00321 l4_uint32_t dummy;
00322 l4_cpu_time_t tsc;
00323 __asm__
00324 (
00325 "movl %%edx, %%ecx \n\t"
00326 "mull %3 \n\t"
00327 "movl %%ecx, %%eax \n\t"
00328 "movl %%edx, %%ecx \n\t"
00329 "mull %3 \n\t"
00330 "addl %%ecx, %%eax \n\t"
00331 "adcl $0, %%edx \n\t"
00332 "shld $5, %%eax, %%edx \n\t"
00333 "shll $5, %%eax \n\t"
00334 : "=A" (tsc),
00335 "=&c" (dummy)
00336 : "0" (ns),
00337 "g" (l4_scaler_ns_to_tsc)
00338);
00339 return tsc;
00340 }
00341
00342 L4_INLINE void
00343 l4_busy_wait_ns (l4_uint64_t ns)
00344 {
00345 l4_cpu_time_t stop = l4_rdtsc();
00346 stop += l4_ns_to_tsc(ns);
00347 while (l4_rdtsc() < stop)
00348 ;
00349 }
00350
00351 L4_INLINE void
00352 l4_busy_wait_us (l4_uint64_t us)
00353 {
00354 l4_cpu_time_t stop = l4_rdtsc ();
00355 stop += l4_ns_to_tsc(us*1000ULL);
00356 while (l4_rdtsc() < stop)
00357 ;
00358 }
00359
00360 EXTERN_C_END
00361
00362 #endif /* __l4_rdtsc_h */
00363
00364
00365

```

## 15.17 amd64/l4/util/spin.h File Reference

Spinning for amd64.

```
#include <l4/sys/compiler.h>
Include dependency graph for spin.h:
```



### 15.17.1 Detailed Description

Spinning for amd64.

Definition in file [spin.h](#).

## 15.18 spin.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef __l4util_spin_h
00014 #define __l4util_spin_h
00015
00016 #include <l4/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_CV void l4_spin(int x,int y);
00021 L4_CV void l4_spin_vga(int x,int y);
00022 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00023 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00024
00025 /*****
00026 *
00027 * spin_text() - spinning wheel at the hercules screen. The given text
00028 * must be a text constant, no variables or arrays. Its
00029 * size is determined with the sizeof operator, it's much
00030 * faster than the strlen function.
00031 *
00032 *****/

```

```

00031 * spin_text_vga() - same for vga.
00032 *
00033 *****/
00034 #define l4_spin_text(x, y, text) \
00035 l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00036 #define l4_spin_text_vga(x, y, text) \
00037 l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00038
00039 EXTERN_C_END
00040
00041 #endif

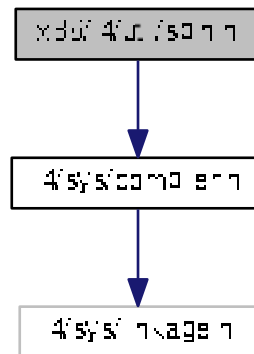
```

## 15.19 x86/l4/util/spin.h File Reference

Spinning for x86.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



### 15.19.1 Detailed Description

Spinning for x86.

Definition in file [spin.h](#).

## 15.20 spin.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef __l4util_spin_h
00014 #define __l4util_spin_h
00015

```

```

00016 #include <14/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_CV void l4_spin(int x,int y);
00021 L4_CV void l4_spin_vga(int x,int y);
00022 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00023 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00024
00025 /*****
00026 *
00027 * spin_text() - spinning wheel at the hercules screen. The given text
00028 * must be a text constant, no variables or arrays. Its
00029 * size is determined with the sizeof operator, it's much
00030 * faster than the strlen function.
00031 * spin_text_vga() - same for vga.
00032 *
00033 *****/
00034 #define l4_spin_text(x, y, text) \
00035 l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00036 #define l4_spin_text_vga(x, y, text) \
00037 l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00038
00039 EXTERN_C_END
00040
00041 #endif

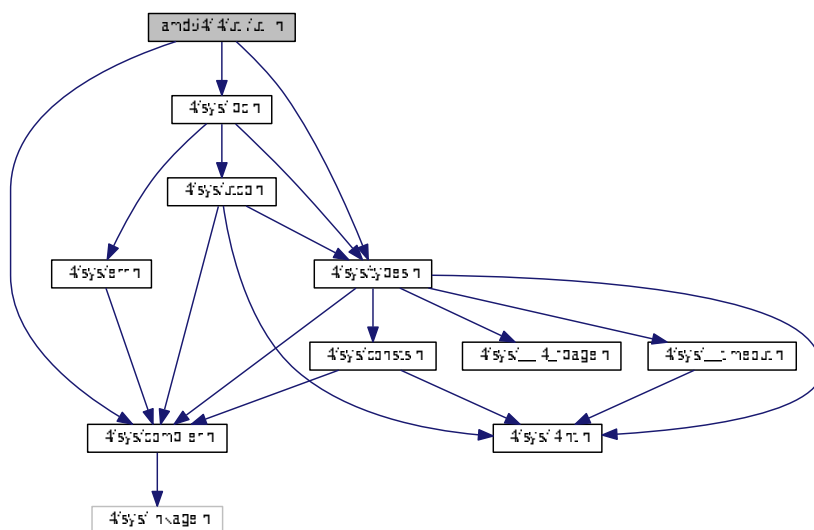
```

## 15.21 amd64/l4/util/util.h File Reference

Utilities, amd64 version.

```
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for util.h:



## Functions

- `l4_timeout_s` `l4util_micros2l4to` (unsigned int mus) `L4_NOTHROW`  
*Calculate l4 timeouts.*

- void [l4\\_sleep](#) (int ms) [L4\\_NOTHROW](#)  
*Suspend thread for a period of ms milliseconds.*
- void [l4\\_sleep\\_forever](#) (void) [L4\\_NOTHROW](#)  
*Go sleep and never wake up.*

### 15.21.1 Detailed Description

Utilities, amd64 version.

Definition in file [util.h](#).

### 15.21.2 Function Documentation

#### 15.21.2.1 l4\_sleep()

```
void l4_sleep (
 int ms)
```

Suspend thread for a period of *ms* milliseconds.

##### Parameters

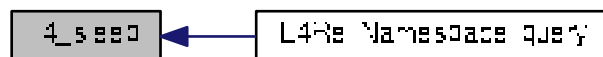
|           |                      |
|-----------|----------------------|
| <i>ms</i> | Time in milliseconds |
|-----------|----------------------|

##### Examples:

[examples/libs/libirq/async\\_isr.c](#), [examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Referenced by [L4Re::Namespace::query\(\)](#).

Here is the caller graph for this function:



## 15.21.2.2 l4util\_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
 unsigned int mus)
```

Calculate l4 timeouts.

## Parameters

|            |                                                                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mus</i> | time in microseconds. Special cases: <ul style="list-style-type: none"> <li>• 0 -&gt; timeout 0</li> <li>• ~0U -&gt; timeout NEVER</li> </ul> |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

the corresponding l4\_timeout value

## 15.22 util.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014 #ifndef __UTIL_H
00015 #define __UTIL_H
00016
00017 #include <l4/sys/types.h>
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/ipc.h>
00020
00021 EXTERN_C_BEGIN
00022
00029 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus)
 L4_NOTHROW;
00030
00034 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00035
00036 /* Suspend thread for a period of \a us microseconds.
00037 * \param us Time in microseconds
00038 * WARNING: This function is mostly bogus since the timer resolution of
00039 * current L4 implementations is about 1ms! */
00040 L4_CV void l4_usleep(int us) L4_NOTHROW;
00041
00047 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00048
00049 L4_INLINE void
00050 l4_sleep_forever(void) L4_NOTHROW
00051 {
00052 for (;;)
00053 l4_ipc_sleep(L4_IPC_NEVER);
00054 }
00055
00057 static inline void
00058 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00059 {
00060 const char *bptr, *eptr;
00061
00062 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00063 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00064 for(;bptr<=eptr;bptr+=L4_PAGE_SIZE){
00065 asm volatile("or %0,%rax \n"
```





- void [l4\\_sleep](#) (int *ms*) [L4\\_NOTHROW](#)  
*Suspend thread for a period of *ms* milliseconds.*
- void [l4\\_sleep\\_forever](#) (void) [L4\\_NOTHROW](#)  
*Go sleep and never wake up.*

### 15.23.1 Detailed Description

Utilities for x86.

Definition in file [util.h](#).

### 15.23.2 Function Documentation

#### 15.23.2.1 l4\_sleep()

```
void l4_sleep (
 int ms)
```

Suspend thread for a period of *ms* milliseconds.

##### Parameters

|           |                      |
|-----------|----------------------|
| <i>ms</i> | Time in milliseconds |
|-----------|----------------------|

#### 15.23.2.2 l4util\_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
 unsigned int mus)
```

Calculate l4 timeouts.

##### Parameters

|            |                                                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mus</i> | time in microseconds. Special cases: <ul style="list-style-type: none"><li>• 0 -&gt; timeout 0</li><li>• ~0U -&gt; timeout NEVER</li></ul> |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|

##### Returns

the corresponding [l4\\_timeout](#) value

## 15.24 util.h

```

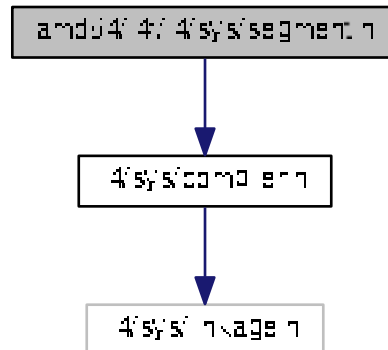
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00009 * Jork Löser <jork@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015 #ifndef __UTIL_H
00016 #define __UTIL_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/ipc.h>
00021
00022 EXTERN_C_BEGIN
00023
00030 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus)
00031 L4_NOTHROW;
00032
00033 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00034
00035 /* Suspend thread for a period of \a us microseconds.
00036 * \param us Time in microseconds
00037 * WARNING: This function is mostly bogus since the timer resolution of
00038 * current L4 implementations is about 1ms! */
00039 L4_CV void l4_usleep(int us) L4_NOTHROW;
00040
00041 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00042
00043 L4_INLINE void
00044 l4_sleep_forever(void) L4_NOTHROW
00045 {
00046 for (;;)
00047 l4_ipc_sleep(L4_IPC_NEVER);
00048 }
00049
00050 static inline void
00051 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00052 {
00053 const char *bptr, *eptr;
00054
00055 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00056 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00057 for(; bptr<=eptr; bptr+=L4_PAGESIZE){
00058 asm volatile("or %0,%eax \n"
00059 :
00060 : "m" (*(const unsigned*)bptr)
00061 : "eax");
00062 }
00063 }
00064
00065 static inline void
00066 l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
00067 {
00068 const char *bptr, *eptr;
00069
00070 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00071 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00072 for(; bptr<=eptr; bptr+=L4_PAGESIZE){
00073 asm volatile("orb $0,%0 \n"
00074 :
00075 : "m" (*(const unsigned*)bptr)
00076 :);
00077 }
00078 }
00079
00080 EXTERN_C_END
00081
00082 #endif
00083

```

## 15.25 amd64/l4f/l4/sys/segment.h File Reference

l4f specific fs/gs manipulation

#include <l4/sys/compiler.h>  
 Include dependency graph for segment.h:



## Functions

- long [fiasco\\_amd64\\_set\\_fs](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the FS register.*
- long [fiasco\\_amd64\\_set\\_segment\\_base](#) ([l4\\_cap\\_idx\\_t](#) thread, enum L4\_sys\_segment segr, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the FS register.*
- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set GDT segment descriptors.*

### 15.25.1 Detailed Description

l4f specific fs/gs manipulation

Definition in file [segment.h](#).

### 15.25.2 Function Documentation

#### 15.25.2.1 fiasco\_amd64\_set\_fs()

```

long fiasco_amd64_set_fs (
 l4_cap_idx_t thread,
 l4_umword_t base,
 l4_utcb_t * utcb) [inline]

```

Set the FS register.

**Parameters**

|               |                          |
|---------------|--------------------------|
| <i>thread</i> | Thread to get info from. |
| <i>base</i>   | Base address.            |
| <i>utcb</i>   | UTCB of the caller.      |

**Returns**

System call error

Definition at line 35 of file [segment.h](#).

**15.25.2.2 fiasco\_amd64\_set\_segment\_base()**

```
long fiasco_amd64_set_segment_base (
 l4_cap_idx_t thread,
 enum L4_sys_segment segr,
 l4_umword_t base,
 l4_utcb_t * utcb) [inline]
```

Set the FS register.

**Parameters**

|               |                                                  |
|---------------|--------------------------------------------------|
| <i>thread</i> | Thread to get info from.                         |
| <i>segr</i>   | Segment register to set (one of L4_sys_segment). |
| <i>base</i>   | Base address.                                    |
| <i>utcb</i>   | UTCB of the caller.                              |

**Returns**

System call error

Definition at line 43 of file [segment.h](#).

**15.26 segment.h**

```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009 * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
```

```

00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00027
00028 #include <l4/sys/compiler.h>
00029
00030 /***** Implementation *****/
00031
00032 *****/
00033
00034 L4_INLINE long
00035 fiasco_amd64_set_fs(l4_cap_idx_t thread,
00036 l4_umword_t base, l4_utcb_t *utcb)
00037 {
00038 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((
00039 l4_umword_t)L4_AMD64_SEGMENT_FS << 16);
00040 l4_utcb_mr_u(utcb)->mr[1] = base;
00041 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00042 L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER), utcb);
00043 }
00044
00045 L4_INLINE long
00046 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum
00047 L4_sys_segment sgr,
00048 l4_umword_t base, l4_utcb_t *utcb)
00049 {
00050 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((
00051 l4_umword_t)sgr << 16);
00052 l4_utcb_mr_u(utcb)->mr[1] = base;
00053 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00054 L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER), utcb);
00055 }
00056
00057 L4_INLINE long
00058 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00059 unsigned int entry_number_start, l4_utcb_t *utcb)
00060 {
00061 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00062 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00063 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00064 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00065 L4_PROTO_THREAD, 2 + (size / 8), 0, 0), L4_IPC_NEVER), utcb);
00066 }
00067
00068 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */

```

## 15.27 amd64/l4/sys/segment.h File Reference

Segment handling.

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```



## 15.27.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

## 15.27.2 Enumeration Type Documentation

### 15.27.2.1 L4\_task\_ldt\_x86\_consts

enum [L4\\_task\\_ldt\\_x86\\_consts](#)

Constants for LDT handling.

#### Enumerator

|                                             |                                                                  |
|---------------------------------------------|------------------------------------------------------------------|
| <a href="#">L4_TASK_LDT_X86_ENTRY_SIZE</a>  | Size of an LDT entry.                                            |
| <a href="#">L4_TASK_LDT_X86_MAX_ENTRIES</a> | Maximum number of LDT entries that can be written with one call. |
| <a href="#">L4_TASK_LDT_X86_ENTRY_SIZE</a>  | Size of an LDT entry.                                            |
| <a href="#">L4_TASK_LDT_X86_MAX_ENTRIES</a> | Maximum number of LDT entries that can be written with one call. |

Definition at line 76 of file [segment.h](#).

## 15.27.3 Function Documentation

### 15.27.3.1 fiasco\_amd64\_segment\_info()

```
long fiasco_amd64_segment_info (
 l4_cap_idx_t thread,
 unsigned * user_ds,
 unsigned * user_cs,
 unsigned * user32_cs,
 l4_utcb_t * utcb) [inline]
```

Get segment information.

#### Parameters

|     |                  |                             |
|-----|------------------|-----------------------------|
| in  | <i>thread</i>    | Thread to get info from.    |
| out | <i>user_ds</i>   | DS segment selector.        |
| out | <i>user_cs</i>   | 64-bit CS segment selector. |
| out | <i>user32_cs</i> | 32-bit CS segment selector. |
| in  | <i>utcb</i>      | UTCB of the caller.         |

**Returns**

System call error

Definition at line 158 of file [segment.h](#).

**15.27.3.2 fiasco\_amd64\_set\_fs()**

```
long fiasco_amd64_set_fs (
 l4_cap_idx_t thread,
 l4_umword_t base,
 l4_utcb_t * utcb) [inline]
```

Set the FS register.

**Parameters**

|               |                          |
|---------------|--------------------------|
| <i>thread</i> | Thread to get info from. |
| <i>base</i>   | Base address.            |
| <i>utcb</i>   | UTCB of the caller.      |

**Returns**

System call error

Definition at line 35 of file [segment.h](#).

**15.27.3.3 fiasco\_amd64\_set\_segment\_base()**

```
long fiasco_amd64_set_segment_base (
 l4_cap_idx_t thread,
 enum L4_sys_segment segr,
 l4_umword_t base,
 l4_utcb_t * utcb) [inline]
```

Set the FS register.

**Parameters**

|               |                                                  |
|---------------|--------------------------------------------------|
| <i>thread</i> | Thread to get info from.                         |
| <i>segr</i>   | Segment register to set (one of L4_sys_segment). |
| <i>base</i>   | Base address.                                    |
| <i>utcb</i>   | UTCB of the caller.                              |



## Returns

System call error

Definition at line 43 of file [segment.h](#).

## 15.28 segment.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 /*****
00024 #ifndef __L4_SYS_ARCH_X86__SEGMENT_H__
00025 #define __L4_SYS_ARCH_X86__SEGMENT_H__
00026
00027 #ifndef L4API_l4f
00028 #error This header file can only be used with a L4API version!
00029 #endif
00030
00031 #include <l4/sys/ipc.h>
00032
00043 L4_INLINE long
00044 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00045 unsigned int entry_number_start, l4_utcb_t *utcb);
00046
00060 L4_INLINE long
00061 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00062 unsigned int entry_number_start, l4_utcb_t *utcb);
00063
00070 L4_INLINE unsigned
00071 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00072 l4_utcb_t *utcb);
00072
00076 enum L4_task_ldt_x86_consts
00077 {
00079 L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00081 L4_TASK_LDT_X86_MAX_ENTRIES
00082 = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00083 / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00084 };
00085
00093 L4_INLINE long
00094 fiasco_amd64_set_fs(l4_cap_idx_t thread,
00095 l4_umword_t base, l4_utcb_t *utcb);
00095
00096 enum L4_sys_segment
00097 {
00098 L4_AMD64_SEGMENT_FS = 0,
00099 L4_AMD64_SEGMENT_GS = 1
00100 };
00101
00110 L4_INLINE long
00111 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum
00112 L4_sys_segment segr,
00113 l4_umword_t base, l4_utcb_t *utcb);
00113
00123 L4_INLINE long
00124 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00125 unsigned *user_cs, unsigned *user32_cs,
00126 l4_utcb_t *utcb);
00126
00127
00128 /*****
00129 *** Implementation
00130 *****/
00131

```

```

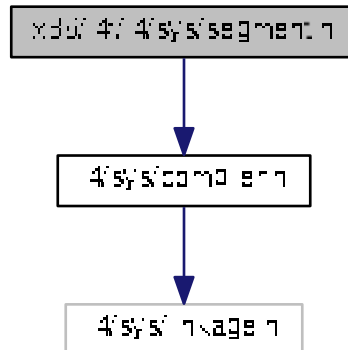
00132 #include <l4/sys/task.h>
00133 #include <l4/sys/thread.h>
00134
00135 L4_INLINE long
00136 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00137 unsigned int entry_number_start, l4_utcb_t *utcb)
00138 {
00139 if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00140 return -L4_EINVAL;
00141 l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00142 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00143 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00144 num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00145 return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(
00146 L4_PROTO_TASK, 2 + num_desc * 2, 0, 0), L4_IPC_NEVER), utcb);
00147 }
00148 L4_INLINE unsigned
00149 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00150 l4_utcb_t *utcb)
00151 {
00152 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00153 if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00154 L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00155 return -1;
00156 return l4_utcb_mr_u(utcb)->mr[0];
00157 }
00158 L4_INLINE long
00159 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00160 unsigned *user_cs, unsigned *user32_cs,
00161 l4_utcb_t *utcb)
00162 {
00163 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00164 int r;
00165 m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;
00166 r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00167 L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb);
00168 if (r < 0)
00169 return r;
00170 *user_ds = m->mr[0];
00171 *user_cs = m->mr[1];
00172 *user32_cs = m->mr[2];
00173 return 0;
00174 }
00175 #endif /* ! __L4_SYS_ARCH_X86_SEGMENT_H__ */

```

## 15.29 x86/i4f/l4/sys/segment.h File Reference

i4f specific segment manipulation

```
#include <l4/sys/compiler.h>
Include dependency graph for segment.h:
```



## Functions

- `long fiasco_gdt_set (l4_cap_idx_t thread, void *desc, unsigned int size, unsigned int entry_number_start, l4_utcb_t *utcb)`  
*Set GDT segment descriptors.*

### 15.29.1 Detailed Description

l4f specific segment manipulation

Definition in file [segment.h](#).

## 15.30 segment.h

```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009 * (c) 2008–2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00027
00028 #include <l4/sys/compiler.h>
```

```

00029
00030 /*****
00031 *** Implementation
00032 *****/
00033
00034 L4_INLINE long
00035 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00036 unsigned int entry_number_start, l4_utcb_t *utcb)
00037 {
00038 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00039 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00040 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00041 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00042 L4_PROTO_THREAD, 2 + (size >> 2), 0, 0), L4_IPC_NEVER), utcb);
00043 }
00044 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */

```

## 15.31 x86/l4/sys/segment.h File Reference

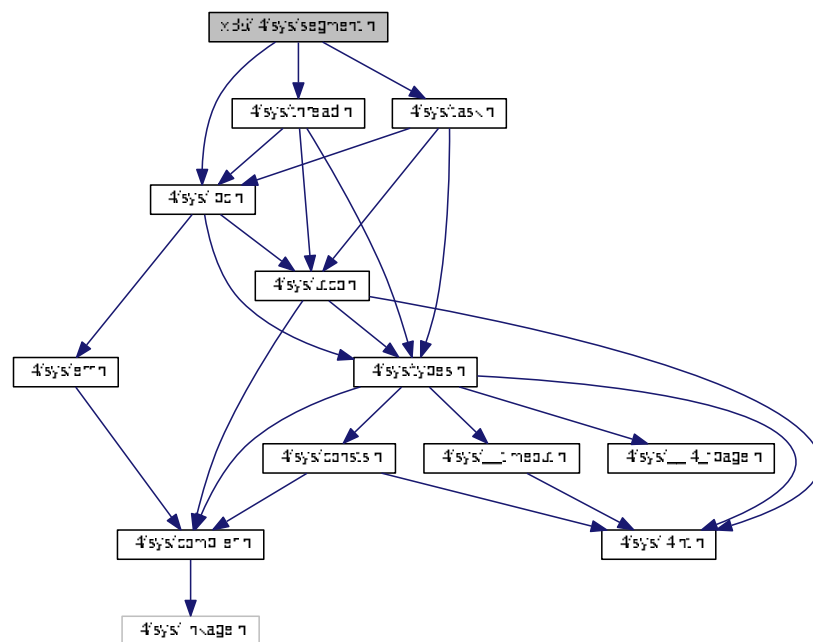
Segment handling.

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```

Include dependency graph for segment.h:



## Enumerations

- enum `L4_task_ldt_x86_consts` { `L4_TASK_LDT_X86_ENTRY_SIZE` = 8, `L4_TASK_LDT_X86_MAX_ENTRIES`, `L4_TASK_LDT_X86_ENTRY_SIZE` = 8, `L4_TASK_LDT_X86_MAX_ENTRIES` }

Constants for LDT handling.

## Functions

- long `fiasco_ldt_set` (`l4_cap_idx_t` task, void \*ldt, unsigned int num\_desc, unsigned int entry\_number\_start, `l4_utcb_t` \*utcb)  
Set LDT segments descriptors.
- long `fiasco_gdt_set` (`l4_cap_idx_t` thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, `l4_utcb_t` \*utcb)  
Set GDT segment descriptors.
- unsigned `fiasco_gdt_get_entry_offset` (`l4_cap_idx_t` thread, `l4_utcb_t` \*utcb)  
Return the offset of the entry in the GDT.

### 15.31.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

### 15.31.2 Enumeration Type Documentation

#### 15.31.2.1 L4\_task\_ldt\_x86\_consts

```
enum L4_task_ldt_x86_consts
```

Constants for LDT handling.

#### Enumerator

|                             |                                                                  |
|-----------------------------|------------------------------------------------------------------|
| L4_TASK_LDT_X86_ENTRY_SIZE  | Size of an LDT entry.                                            |
| L4_TASK_LDT_X86_MAX_ENTRIES | Maximum number of LDT entries that can be written with one call. |
| L4_TASK_LDT_X86_ENTRY_SIZE  | Size of an LDT entry.                                            |
| L4_TASK_LDT_X86_MAX_ENTRIES | Maximum number of LDT entries that can be written with one call. |

Definition at line 76 of file [segment.h](#).

## 15.32 segment.h

```
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
```

```

00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 /*****
00024 #ifndef __L4_SYS__ARCH_X86__SEGMENT_H__
00025 #define __L4_SYS__ARCH_X86__SEGMENT_H__
00026
00027 #ifndef L4API_l4f
00028 #error This header file can only be used with a L4API version!
00029 #endif
00030
00031 #include <l4/sys/ipc.h>
00032
00043 L4_INLINE long
00044 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int size,
00045 unsigned int entry_number_start, l4_utcb_t *utcb);
00046
00060 L4_INLINE long
00061 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00062 unsigned int entry_number_start, l4_utcb_t *utcb);
00063
00070 L4_INLINE unsigned
00071 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00072 l4_utcb_t *utcb);
00072
00076 enum L4_task_ldt_x86_consts
00077 {
00079 L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00081 L4_TASK_LDT_X86_MAX_ENTRIES
00082 = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00083 / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00084 };
00085
00086 /*****
00087 *** Implementation
00088 *****/
00089
00090 #include <l4/sys/task.h>
00091 #include <l4/sys/thread.h>
00092
00093 L4_INLINE long
00094 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00095 unsigned int entry_number_start, l4_utcb_t *utcb)
00096 {
00097 if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00098 return -L4_EINVAL;
00099 l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00100 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00101 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00102 num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00103 return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(
00104 L4_PROTO_TASK, 2 + num_desc * 2, 0, 0), L4_IPC_NEVER), utcb);
00104 }
00105
00106 L4_INLINE unsigned
00107 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00108 l4_utcb_t *utcb)
00109 {
00109 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00110 if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00111 L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00111 return -1;
00112 return l4_utcb_mr_u(utcb)->mr[0];
00113 }
00114
00115 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */

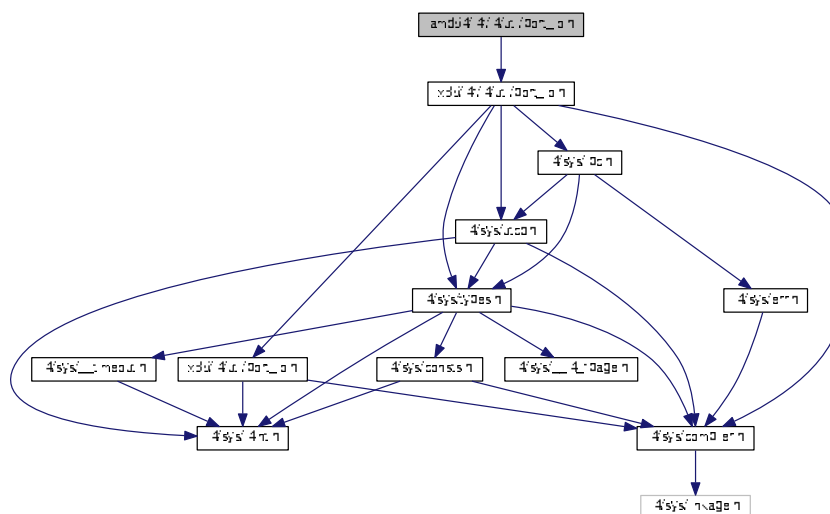
```

## 15.33 amd64/l4/l4/util/port\_io.h File Reference

Port I/O functions.

```
#include <x86/l4f/l4/util/port_io.h>
```

Include dependency graph for port io.h:



### 15.33.1 Detailed Description

## Port I/O functions.

Definition in file [port\\_io.h](#).

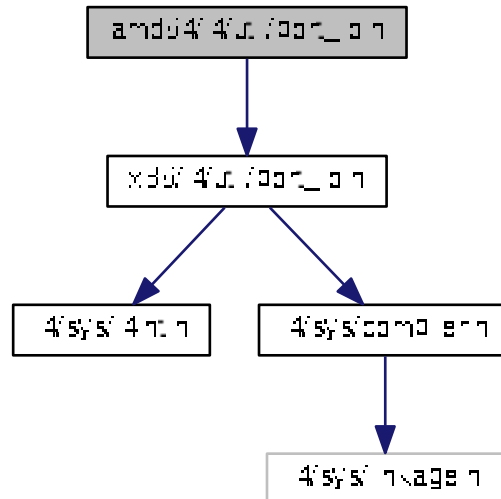
### 15.34 port\_io.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #include <x86/14f/14/util/port_io.h>
```

### 15.35 amd64/l4/util/port\_io.h File Reference

## Port I/O functions.

```
#include <x86/l4/util/port_io.h>
Include dependency graph for port_io.h:
```



### 15.35.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 15.36 port\_io.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #include <x86/l4/util/port_io.h>
```

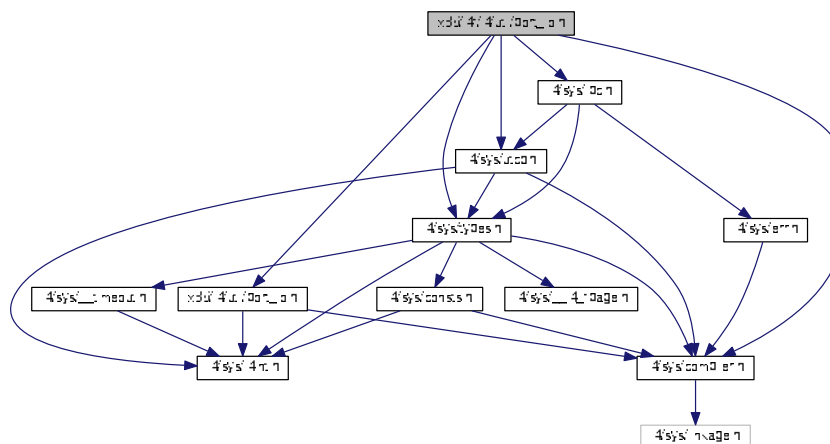
## 15.37 x86/l4/l4/util/port\_io.h File Reference

port I/O functions

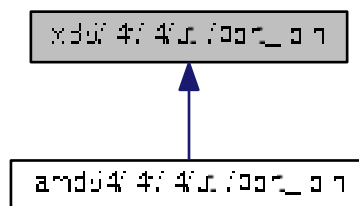
```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <x86/l4/util/port_io.h>
#include <l4/sys/utcb.h>
```



```
#include <linux/sys/ipc.h>
```



This graph shows which files directly or indirectly include this file:



## Functions

- `int l4util_ioport_map (l4_cap_idx_t sigma0id, unsigned port_start, unsigned log2size)`  
*Map a range of I/O ports.*

### 15.37.1 Detailed Description

## port I/O functions

Date \_\_\_\_\_

06/2003

## Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 15.37.2 Function Documentation

### 15.37.2.1 l4util\_ioport\_map()

```
int l4util_ioport_map (
 l4_cap_idx_t sigma0id,
 unsigned port_start,
 unsigned log2size) [inline]
```

Map a range of I/O ports.

#### Parameters

|                   |                               |
|-------------------|-------------------------------|
| <i>sigma0id</i>   | I/O port service (sigma0).    |
| <i>port_start</i> | (Start) Port to request.      |
| <i>log2size</i>   | Log2size of range to request. |

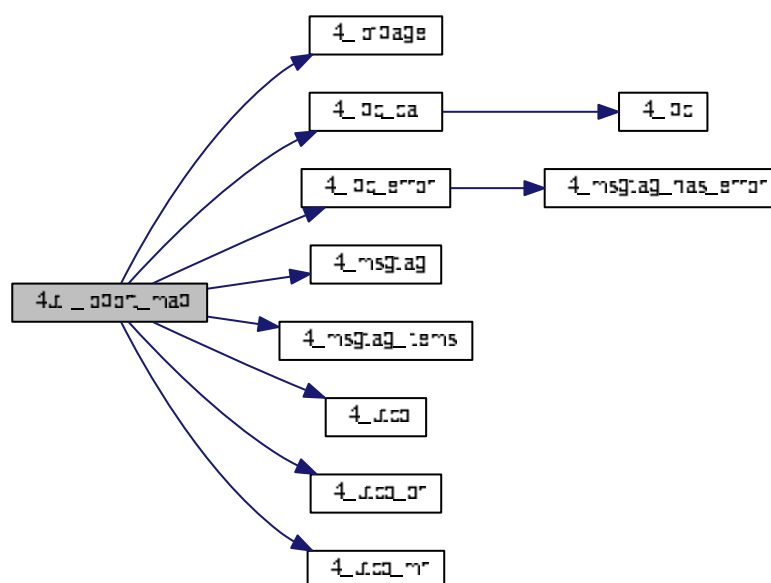
#### Returns

IPC result: 0 if the range could be successfully mapped on error: IPC failure, or -L4\_ENOENT if nothing mapped

Definition at line 56 of file [port\\_io.h](#).

References [l4\\_buf\\_regs\\_t::bdr](#), [l4\\_buf\\_regs\\_t::br](#), [L4\\_ENOENT](#), [l4\\_iofpage\(\)](#), [l4\\_ipc\\_call\(\)](#), [l4\\_ipc\\_error\(\)](#), [L4\\_IPC\\_NEVER](#), [L4\\_ITEM\\_MAP](#), [l4\\_msgtag\(\)](#), [l4\\_msgtag\\_items\(\)](#), [L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#), [l4\\_utcb\(\)](#), [l4\\_utcb\\_br\(\)](#), [l4\\_utcb\\_mr\(\)](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_fpage\\_t::raw](#).

Here is the call graph for this function:



## 15.38 port\_io.h

```

00001 /*****/
00009 /*****/
00010
00011 /*
00012 * (c) 2003-2009 Author(s)
00013 * economic rights: Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 #ifndef _L4UTIL_PORT_IO_API_H
00020 #define _L4UTIL_PORT_IO_API_H
00021
00022 #include <l4/sys/compiler.h>
00023 #include <l4/sys/types.h>
00024
00025 #include <x86/l4/util/port_io.h>
00026
00027 EXTERN_C_BEGIN
00028
00029 L4_INLINE int
00041 l4util_ioport_map(l4_cap_idx_t sigma0id,
00042 unsigned port_start, unsigned log2size);
00043
00044 EXTERN_C_END
00045
00046
00047 /*****/
00048 *** Implementation
00049 *****/
00050
00051 #include <l4/sys/utcb.h>
00052 #include <l4/sys/ipc.h>
00053
00054
00055 L4_INLINE int
00056 l4util_ioport_map(l4_cap_idx_t sigma0id,
00057 unsigned port_start, unsigned log2size)
00058 {
00059 l4_fpage_t iofp;
00060 l4_msgtag_t tag;
00061 long err;
00062
00063 iofp = l4_iofpage(port_start, log2size);
00064 l4_utcb_mr()->mr[0] = iofp.raw;
00065 l4_utcb_br()->bdr = 0;
00066 l4_utcb_br()->br[0] = L4_ITEM_MAP;
00067 l4_utcb_br()->br[1] = iofp.raw;
00068 tag = l4_ipc_call(sigma0id, l4_utcb(),
00069 l4_msgtag(L4_PROTO_IO_PAGE_FAULT, 1, 0, 0),
00070 L4_IPC_NEVER);
00071
00072 if ((err = l4_ipc_error(tag, l4_utcb())))
00073 return err;
00074
00075 return l4_msgtag_items(tag) > 0 ? 0 : -L4_ENOENT;
00076 }
00077
00078 #endif
00079

```

## 15.39 x86/l4/util/port\_io.h File Reference

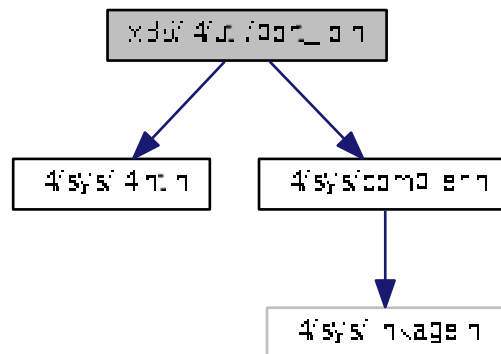
x86 port I/O

```

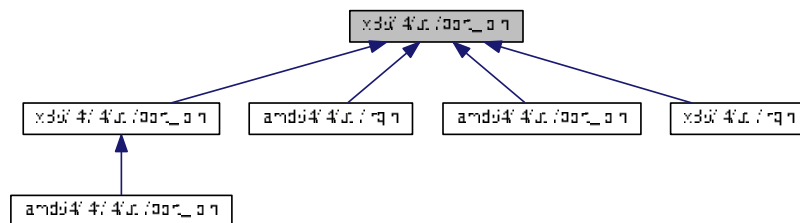
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for `port_io.h`:



This graph shows which files directly or indirectly include this file:



## Functions

- `l4_uint8_t l4util_in8 (l4_uint16_t port)`  
Read byte from I/O port.
- `l4_uint16_t l4util_in16 (l4_uint16_t port)`  
Read 16-bit-value from I/O port.
- `l4_uint32_t l4util_in32 (l4_uint16_t port)`  
Read 32-bit-value from I/O port.
- `void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
Read a block of 8-bit-values from I/O ports.
- `void l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
Read a block of 16-bit-values from I/O ports.
- `void l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
Read a block of 32-bit-values from I/O ports.
- `void l4util_out8 (l4_uint8_t value, l4_uint16_t port)`  
Write byte to I/O port.
- `void l4util_out16 (l4_uint16_t value, l4_uint16_t port)`

*Write 16-bit-value to I/O port.*

- void `l4util_out32` (`l4_uint32_t` value, `l4_uint16_t` port)

*Write 32-bit-value to I/O port.*

- void `l4util_outs8` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)

*Write a block of bytes to I/O port.*

- void `l4util_outs16` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)

*Write a block of 16-bit-values to I/O port.*

- void `l4util_outs32` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)

*Write block of 32-bit-values to I/O port.*

- void `l4util_iodelay` (void)

*delay I/O port access by writing to port 0x80*

## 15.39.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 15.40 port\_io.h

```
00001 /*****
00009 /*****
00010
00011 /*
00012 * (c) 2003-2009 Author(s)
00013 * economic rights; Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 #ifndef _L4UTIL_PORT_IO_H
00020 #define _L4UTIL_PORT_IO_H
00021
00022 /* L4 includes */
00023 #include <l4/sys/l4int.h>
00024 #include <l4/sys/compiler.h>
00025
00026 /*****
00027 *** Prototypes
00028 *****/
00029
00030 EXTERN_C_BEGIN
00031 L4_INLINE l4_uint8_t
00032 l4util_in8(l4_uint16_t port);
00033
00034 L4_INLINE l4_uint16_t
00035 l4util_in16(l4_uint16_t port);
00036
00037 L4_INLINE l4_uint32_t
00038 l4util_in32(l4_uint16_t port);
00039
00040 L4_INLINE void
00041 l4util_ins8(l4_uint16_t port, l4_umword_t addr,
```

```

 l4_umword_t count);
00076
00084 L4_INLINE void
00085 l4util_ins16(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00086
00094 L4_INLINE void
00095 l4util_ins32(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00096
00103 L4_INLINE void
00104 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00105
00113 L4_INLINE void
00114 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00115
00122 L4_INLINE void
00123 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00124
00132 L4_INLINE void
00133 l4util_outs8(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00134
00143 L4_INLINE void
00144 l4util_outs16(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00145
00153 L4_INLINE void
00154 l4util_outs32(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00155
00159 L4_INLINE void
00160 l4util_iodelay(void);
00161
00164 EXTERN_C_END
00165
00166
00167 /*****
00168 *** Implementation
00169 *****/
00170
00171 L4_INLINE l4_uint8_t
00172 l4util_in8(l4_uint16_t port)
00173 {
00174 l4_uint8_t value;
00175 asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
00176 return value;
00177 }
00178
00179 L4_INLINE l4_uint16_t
00180 l4util_in16(l4_uint16_t port)
00181 {
00182 l4_uint16_t value;
00183 asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
00184 return value;
00185 }
00186
00187 L4_INLINE l4_uint32_t
00188 l4util_in32(l4_uint16_t port)
00189 {
00190 l4_uint32_t value;
00191 asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
00192 return value;
00193 }
00194
00195 L4_INLINE void
00196 l4util_ins8(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count)
00197 {
00198 l4_umword_t dummy1, dummy2;
00199 asm volatile ("rep insb" : "=D" (dummy1), "=c" (dummy2)
00200 : "d" (port), "D" (addr), "c" (count)
00201 : "memory");
00202 }
00203
00204 L4_INLINE void
00205 l4util_ins16(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count)
00206 {
00207 l4_umword_t dummy1, dummy2;
00208 asm volatile ("rep insw" : "=D" (dummy1), "=c" (dummy2)
00209 : "d" (port), "D" (addr), "c" (count)
00210 : "memory");
00211 }
00212
00213 L4_INLINE void
00214 l4util_ins32(l4_uint16_t port, l4_umword_t addr,

```

```

 l4_umword_t count)
00215 {
00216 l4_umword_t dummy1, dummy2;
00217 asm volatile ("rep insl" : "=D"(dummy1), "=c"(dummy2)
00218 : "d" (port), "D" (addr), "c"(count)
00219 : "memory");
00220 }
00221
00222 L4_INLINE void
00223 l4util_out8(l4_uint8_t value, l4_uint16_t port)
00224 {
00225 asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
00226 }
00227
00228 L4_INLINE void
00229 l4util_out16(l4_uint16_t value, l4_uint16_t port)
00230 {
00231 asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
00232 }
00233
00234 L4_INLINE void
00235 l4util_out32(l4_uint32_t value, l4_uint16_t port)
00236 {
00237 asm volatile ("outl %0, %w1" : : "a" (value), "Nd" (port));
00238 }
00239
00240 L4_INLINE void
00241 l4util_outs8(l4_uint16_t port, l4_umword_t addr,
00242 l4_umword_t count)
00243 {
00244 l4_umword_t dummy1, dummy2;
00245 asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
00246 : "d" (port), "S" (addr), "c"(count)
00247 : "memory");
00248 }
00249
00250 L4_INLINE void
00251 l4util_outs16(l4_uint16_t port, l4_umword_t addr,
00252 l4_umword_t count)
00253 {
00254 l4_umword_t dummy1, dummy2;
00255 asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
00256 : "d" (port), "S" (addr), "c"(count)
00257 : "memory");
00258 }
00259
00260 L4_INLINE void
00261 l4util_outs32(l4_uint16_t port, l4_umword_t addr,
00262 l4_umword_t count)
00263 {
00264 l4_umword_t dummy1, dummy2;
00265 asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
00266 : "d" (port), "S" (addr), "c"(count)
00267 : "memory");
00268 }
00269
00270 L4_INLINE void
00271 l4util_iodelay(void)
00272 {
00273 asm volatile ("outb %a1, $0x80");
00274 }
00275 #endif

```

## 15.41 amd64/l4f/l4/util/setjmp.h File Reference

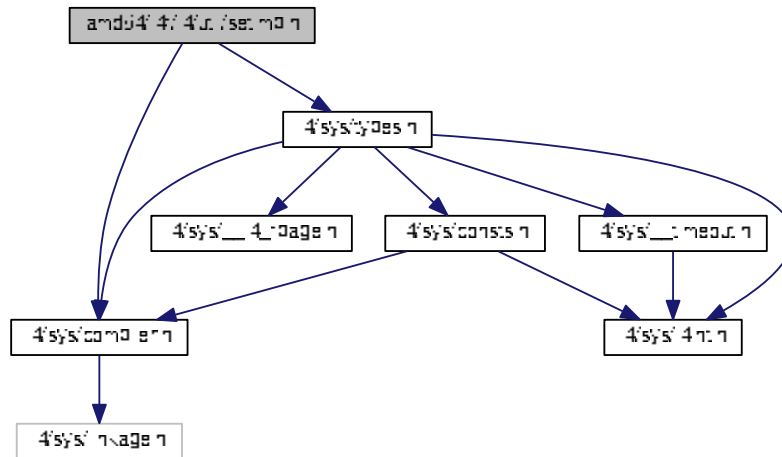
Inter-thread setjmp/longjmp.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for `setjmp.h`:



## Functions

- `int l4_thread_setjmp (l4_thread_jump_buf env)`  
*inter-thread setjmp*
- `void l4_thread_longjmp (l4_threadid_t thread, l4_thread_jump_buf env, int val)`  
*inter-thread longjmp*

### 15.41.1 Detailed Description

Inter-thread `setjmp/longjmp`.

Date

12/21/2005

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [setjmp.h](#).

### 15.41.2 Function Documentation

#### 15.41.2.1 l4\_thread\_longjmp()

```
void l4_thread_longjmp (
 l4_threadid_t thread,
 l4_thread_jump_buf env,
 int val)
```

inter-thread `longjmp`

This function sets `thread` to the location obtained by its former `l4_thread_setjump` on `env`.



## Parameters

|               |                                |
|---------------|--------------------------------|
| <i>thread</i> | thread to apply the longjmp to |
| <i>env</i>    | jump buffer                    |
| <i>val</i>    | 0: setjmp returns with 1       |
| <i>val</i>    | !0: return value of setjmp     |

## See also

longjmp(3)

## Note

In contrast to longjmp(3), this function returns.

## 15.41.2.2 l4\_thread\_setjmp()

```
int l4_thread_setjmp (
 l4_thread_jump_buf env)
```

## inter-thread setjmp

Use this function to prepare a longjmp from another thread for this thread.

## Parameters

|            |             |
|------------|-------------|
| <i>env</i> | jump buffer |
|------------|-------------|

## Return values

|    |                       |
|----|-----------------------|
| 0  | returned directly     |
| !0 | returned from longjmp |

## See also

setjmp(3)

## 15.42 setjmp.h

```
00001
00009 /*
00010 * (c) 2004-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __UTIL_INCLUDE_ARCH_AMD64_L4API_L4F_SETJMP_H_
```

```

00017 #define __UTIL_INCLUDE_ARCH_Amd64_L4API_L4F_SETJMP_H_
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 typedef struct{
00024 l4_umword_t r8; /* 0x00 */
00025 l4_umword_t r9; /* 0x08 */
00026 l4_umword_t r10; /* 0x10 */
00027 l4_umword_t r11; /* 0x18 */
00028 l4_umword_t r12; /* 0x20 */
00029 l4_umword_t r13; /* 0x28 */
00030 l4_umword_t r14; /* 0x30 */
00031 l4_umword_t r15; /* 0x38 */
00032 l4_umword_t rbx; /* 0x40 */
00033 l4_umword_t rsi; /* 0x48 */
00034 l4_umword_t rbp; /* 0x50 */
00035 l4_umword_t rsp; /* 0x58 */
00036 l4_umword_t rip; /* 0x60 */
00037 l4_umword_t rip_caller; /* 0x68 */
00038 l4_umword_t rflags; /* 0x70 */
00039 l4_umword_t stack[40];
00040 } l4_thread_jump_buf_s;
00041 typedef int l4_thread_jump_buf[sizeof(l4_thread_jump_buf_s)/sizeof(l4_umword_t)];
00042
00043 typedef union{
00044 l4_thread_jump_buf_s s;
00045 l4_thread_jump_buf raw;
00046 } l4_thread_jump_buf_u;
00047
00059 L4_CV int l4_thread_setjmp(l4_thread_jump_buf env);
00060
00075 L4_CV void l4_thread_longjmp(l4_threadid_t thread, l4_thread_jump_buf env, int val);
00076
00077 EXTERN_C_END
00078
00079 #endif

```

## 15.43 x86/i4f/i4/util/setjmp.h File Reference

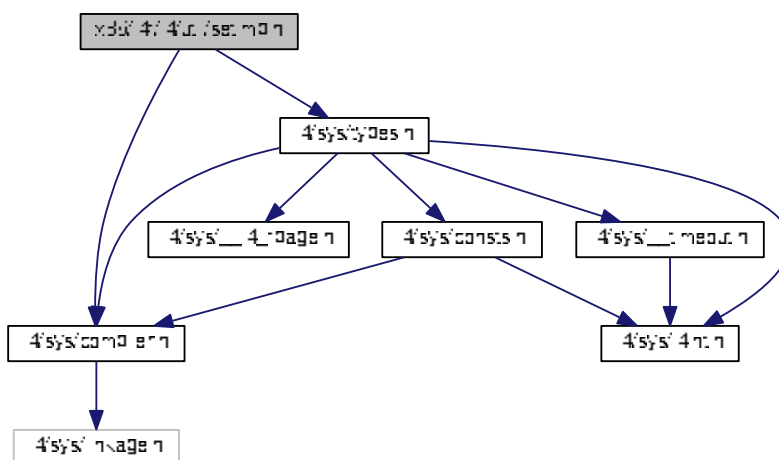
Inter-thread setjmp/longjmp.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for setjmp.h:



## Functions

- int [l4\\_thread\\_setjmp](#) (l4\_thread\_jmp\_buf env)  
*inter-thread setjmp*
- void [l4\\_thread\\_longjmp](#) (l4\_threadid\_t thread, l4\_thread\_jmp\_buf env, int val)  
*inter-thread longjmp*

### 15.43.1 Detailed Description

Inter-thread setjmp/longjmp.

#### Date

11/26/2004

#### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [setjmp.h](#).

### 15.43.2 Function Documentation

#### 15.43.2.1 l4\_thread\_longjmp()

```
void l4_thread_longjmp (
 l4_threadid_t thread,
 l4_thread_jmp_buf env,
 int val)
```

inter-thread longjmp

This function sets `thread` to the location obtained by its former `l4_thread_setjump` on `env`.

#### Parameters

|               |                                |
|---------------|--------------------------------|
| <i>thread</i> | thread to apply the longjmp to |
| <i>env</i>    | jump buffer                    |
| <i>val</i>    | 0: setjmp returns with 1       |
| <i>val</i>    | !0: return value of setjmp     |

#### See also

[longjmp\(3\)](#)

**Note**

In contrast to `longjmp(3)`, this function returns.

**15.43.2.2 l4\_thread\_setjmp()**

```
int l4_thread_setjmp (
 l4_thread_jump_buf env)
```

**inter-thread setjmp**

Use this function to prepare a `longjmp` from another thread for this thread.

**Parameters**

|            |             |
|------------|-------------|
| <i>env</i> | jump buffer |
|------------|-------------|

**Return values**

|    |                                    |
|----|------------------------------------|
| 0  | returned directly                  |
| !0 | returned from <code>longjmp</code> |

**See also**

`setjmp(3)`

**15.44 setjmp.h**

```
00001
00009 /*
00010 * (c) 2004-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __UTIL_INCLUDE_ARCH_X86_L4API_L4F_SETJMP_H_
00017 #define __UTIL_INCLUDE_ARCH_X86_L4API_L4F_SETJMP_H_
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 typedef struct{
00024 l4_umword_t ebx; /* 0 */
00025 l4_umword_t esi; /* 4 */
00026 l4_umword_t edi; /* 8 */
00027 l4_umword_t ebp; /* 12 */
00028 l4_umword_t esp; /* 16 */
00029 l4_umword_t eip; /* 20 */
00030 l4_umword_t eip_caller; /* 24 */
00031 l4_umword_t eflags; /* 28 */
00032 l4_umword_t stack[40];
00033 } l4_thread_jump_buf_s;
00034 typedef int l4_thread_jump_buf[sizeof(l4_thread_jump_buf_s)/sizeof(l4_umword_t)];
00035
00036 typedef union{
00037 l4_thread_jump_buf_s s;
00038 l4_thread_jump_buf raw;
```

```

00039 } l4_thread_jump_buf_u;
00040
00052 extern int l4_thread_setjmp(l4_thread_jump_buf env);
00053
00068 void l4_thread_longjmp(l4_threadid_t thread, l4_thread_jump_buf env, int val);
00069
00070 EXTERN_C_END
00071
00072 #endif

```

## 15.45 arm/l4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV`  
Define calling convention.

### 15.45.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 15.46 linkage.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4_SYS_ARCH_ARM_LINKAGE_H__
00025 #define __L4_SYS_ARCH_ARM_LINKAGE_H__
00026
00027 #ifdef __ASSEMBLY__
00028 #ifndef ENTRY
00029 #define ENTRY(name) \
00030 .globl name; \
00031 .p2align(2); \
00032 name:
00033 #endif
00034 #endif
00035
00036 #define L4_FASTCALL(x) x
00037 #define l4_fastcall
00038
00044 #define L4_CV
00045
00046 #ifdef __PIC__
00047 # define L4_LONG_CALL
00048 #else
00049 # define L4_LONG_CALL __attribute__((long_call))
00050 #endif
00051
00052 #endif /* ! __L4_SYS_ARCH_ARM_LINKAGE_H__ */

```

## 15.47 amd64/l4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV`  
*Define calling convention.*

### 15.47.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 15.48 linkage.h

```

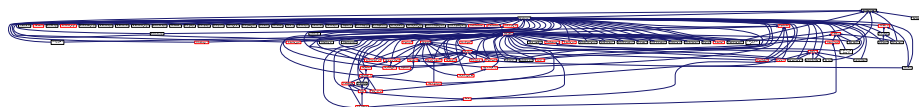
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4__SYS__ARCH_AMD64__LINKAGE_H__
00026 #define __L4__SYS__ARCH_AMD64__LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032 .globl name; \
00033 .p2align(2); \
00034 name:
00035
00036 #endif /* __ASSEMBLY__ */
00037 #endif /* ! ENTRY */
00038
00039 #define L4_FASTCALL(x) x
00040 #define l4_fastcall
00041
00047 #define L4_CV
00048
00049 #endif /* ! __L4__SYS__ARCH_AMD64__LINKAGE_H__ */

```

## 15.49 x86/l4/sys/linkage.h File Reference

Linkage.

This graph shows which files directly or indirectly include this file:



## Macros

- `#define L4_CV __attribute__((regparm(0)))`  
*Define calling convention.*

### 15.49.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 15.50 linkage.h

```

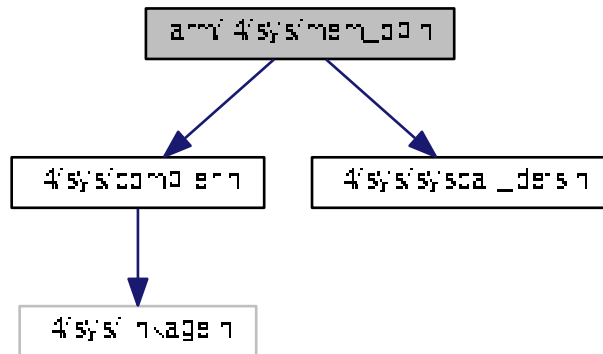
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4__SYS__ARCH_X86__LINKAGE_H__
00026 #define __L4__SYS__ARCH_X86__LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032 .globl name; \
00033 .p2align(2); \
00034 name:
00035
00036 #endif /* ! ENTRY */
00037 #endif /* __ASSEMBLY__ */
00038
00039 #define L4_FASTCALL(x) x __attribute__((regparm(3)))
00040 #define l4_fastcall __attribute__((regparm(3)))
00041
00047 #define L4_CV __attribute__((regparm(0)))
00048
00049 #endif /* ! __L4__SYS__ARCH_X86__LINKAGE_H__ */

```

## 15.51 arm/l4/sys/mem\_op.h File Reference

Memory access functions (ARM specific)

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
Include dependency graph for mem_op.h:
```



## Enumerations

- enum [L4\\_mem\\_op\\_widths](#) { [L4\\_MEM\\_WIDTH\\_1BYTE](#) = 0, [L4\\_MEM\\_WIDTH\\_2BYTE](#) = 1, [L4\\_MEM\\_WIDTH\\_4BYTE](#) = 2 }
- Memory access width definitions.*

## Functions

- unsigned long [l4\\_mem\\_read](#) (unsigned long virtaddress, unsigned width)  
*Read user task memory from kernel privilege level.*
- void [l4\\_mem\\_write](#) (unsigned long virtaddress, unsigned width, unsigned long value)  
*Write user task memory from kernel privilege level.*
- unsigned long [l4\\_mem\\_arm\\_op\\_call](#) (unsigned long op, unsigned long va, unsigned long width, unsigned long value)  
*Implementations.*

### 15.51.1 Detailed Description

Memory access functions (ARM specific)

Date

2010-10

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [mem\\_op.h](#).



## 15.52 mem\_op.h

```

00001
00009 /*
00010 * (c) 2010 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 EXTERN_C_BEGIN
00033
00051 enum L4_mem_op_widths
00052 {
00053 L4_MEM_WIDTH_1BYTE = 0,
00054 L4_MEM_WIDTH_2BYTE = 1,
00055 L4_MEM_WIDTH_4BYTE = 2,
00056 };
00057
00070 L4_INLINE unsigned long
00071 l4_mem_read(unsigned long virtaddress, unsigned width);
00072
00085 L4_INLINE void
00086 l4_mem_write(unsigned long virtaddress, unsigned width,
00087 unsigned long value);
00088
00089 enum L4_mem_ops
00090 {
00091 L4_MEM_OP_MEM_READ = 0x10,
00092 L4_MEM_OP_MEM_WRITE = 0x11,
00093 };
00094
00098 L4_INLINE unsigned long
00099 l4_mem_arm_op_call(unsigned long op,
00100 unsigned long va,
00101 unsigned long width,
00102 unsigned long value);
00103
00106 L4_INLINE unsigned long
00107 l4_mem_arm_op_call(unsigned long op,
00108 unsigned long va,
00109 unsigned long width,
00110 unsigned long value)
00111 {
00112 register unsigned long _op __asm__ ("r0") = op;
00113 register unsigned long _va __asm__ ("r1") = va;
00114 register unsigned long _width __asm__ ("r2") = width;
00115 register unsigned long _value __asm__ ("r3") = value;
00116
00117 __asm__ __volatile__
00118 ("@ l4_cache_op_arm_call(start) \n\t"
00119 "mov lr, pc \n\t"
00120 "mov pc, %[sc] \n\t"
00121 "@ l4_cache_op_arm_call(end) \n\t"
00122 :
00123 "=r" (_op),
00124 "=r" (_va),
00125 "=r" (_width),
00126 "=r" (_value)
00127 :
00128 [sc] "i" (L4_SYSCALL_MEM_OP),
00129 "0" (_op),
00130 "1" (_va),
00131 "2" (_width),
00132 "3" (_value)
00133 :
00134 "cc", "memory", "lr"
00135);
00136
00137 return _value;

```

```

00138 }
00139
00140 L4_INLINE unsigned long
00141 l4_mem_read(unsigned long virtaddress, unsigned width)
00142 {
00143 return l4_mem_arm_op_call(L4_MEM_OP_MEM_READ, virtaddress, width, 0);
00144 }
00145
00146 L4_INLINE void
00147 l4_mem_write(unsigned long virtaddress, unsigned width,
00148 unsigned long value)
00149 {
00150 l4_mem_arm_op_call(L4_MEM_OP_MEM_WRITE, virtaddress, width, value);
00151 }
00152
00153 EXTERN_C_END
00154
00155 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__ */

```

## 15.53 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

### Data Structures

- struct [l4\\_vm\\_tz\\_state](#)  
*state structure for TrustZone VMs*

### 15.53.1 Detailed Description

ARM virtualization interface.

Definition in file [vm.h](#).

## 15.54 vm.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00035 struct l4_vm_tz_state_mode
00036 {
00037 l4_umword_t sp;
00038 l4_umword_t lr;
00039 l4_umword_t spsr;
00040 };
00041
00042 struct l4_vm_tz_state_irq_inject

```

```

00043 {
00044 l4_uint32_t group;
00045 l4_uint32_t irqs[8];
00046 };
00047
00052 struct l4_vm_tz_state
00053 {
00054 l4_umword_t r[13]; // r0 - r12
00055
00056 l4_umword_t sp_usr;
00057 l4_umword_t lr_usr;
00058
00059 struct l4_vm_tz_state_mode irq;
00060
00061 l4_umword_t r_fiq[5]; // r8 - r12
00062 struct l4_vm_tz_state_mode fiq;
00063 struct l4_vm_tz_state_mode abt;
00064 struct l4_vm_tz_state_mode und;
00065 struct l4_vm_tz_state_mode svc;
00066
00067 l4_umword_t pc;
00068 l4_umword_t cpsr;
00069
00070 l4_umword_t pending_events;
00071 l4_uint32_t cpacr;
00072 l4_umword_t cpl0_fpexc;
00073
00074 l4_umword_t pfs;
00075 l4_umword_t pfa;
00076 l4_umword_t exit_reason;
00077
00078 struct l4_vm_tz_state_irq_inject irq_inject;
00079 };
00080
00081 enum L4_vm_exit_reason
00082 {
00083 L4_vm_exit_reason_vmm_call = 1,
00084 L4_vm_exit_reason_inst_abort = 2,
00085 L4_vm_exit_reason_data_abort = 3,
00086 L4_vm_exit_reason_irq = 4,
00087 L4_vm_exit_reason_fiq = 5,
00088 L4_vm_exit_reason_undef = 6,
00089 };
00090
00091 L4_INLINE int
00092 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq);
00093
00094 L4_INLINE int
00095 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq)
00096 {
00097 if (irq > sizeof(state->irq_inject.irqs) * 8)
00098 return -L4_EINVAL;
00099
00100 unsigned g = irq / 32;
00101 state->irq_inject.group |= 1 << g;
00102 state->irq_inject.irqs[g] |= 1 << (irq & 31);
00103
00104 return 0;
00105 }

```

## 15.55 arm/l4/util/bitops\_arch.h File Reference

ARM specific implementation of bitops functions.

### 15.55.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file [bitops\\_arch.h](#).

## 15.56 bitops\_arch.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00013 #define __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00014
00015
00016 #endif /* ! __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__ */

```

## 15.57 amd64/l4/util/bitops\_arch.h File Reference

amd64 bit manipulation functions

### 15.57.1 Detailed Description

amd64 bit manipulation functions

Date

07/03/2001

Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de) Torsten Frenzel [frenzel@os.inf.tu-dresden.de](mailto:frenzel@os.inf.tu-dresden.de)  
 de Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [bitops\\_arch.h](#).

## 15.58 bitops\_arch.h

```

00001 /*****/
00011 /*
00012 * (c) 2000-2009 Author(s)
00013 * economic rights: Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 /*****/
00020 #ifndef __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00021 #define __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00022
00023 EXTERN_C_BEGIN
00024
00025 /*****/
00026 *** Implementation
00027 *****/
00028
00029 #define __L4UTIL__BITOPS_HAVE_ARCH_SET_BIT
00030 L4_INLINE void
00031 l4util_set_bit(int b, volatile l4_umword_t * dest)
00032 {
00033 __asm__ __volatile__

```

```

00034 (
00035 "lock; bts %1,%0 \n\t"
00036 :
00037 :
00038 "m" (*dest), /* 0 mem, destination operand */
00039 "Ir" (b) /* 1, bit number */
00040 :
00041 "memory", "cc"
00042);
00043 }
00044
00045 /* clear bit */
00046 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00047 L4_INLINE void
00048 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00049 {
00050 __asm__ __volatile__
00051 (
00052 "lock; btr %1,%0 \n\t"
00053 :
00054 :
00055 "m" (*dest), /* 0 mem, destination operand */
00056 "Ir" (b) /* 1, bit number */
00057 :
00058 "memory", "cc"
00059);
00060 }
00061
00062 /* change bit */
00063 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00064 L4_INLINE void
00065 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00066 {
00067 __asm__ __volatile__
00068 (
00069 "lock; btc %1,%0 \n\t"
00070 :
00071 :
00072 "m" (*dest), /* 0 mem, destination operand */
00073 "Ir" (b) /* 1, bit number */
00074 :
00075 "memory", "cc"
00076);
00077 }
00078
00079 /* test bit */
00080 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00081 L4_INLINE int
00082 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00083 {
00084 l4_int8_t bit;
00085
00086 __asm__ __volatile__
00087 (
00088 "bt %2,%1 \n\t"
00089 "setc %0 \n\t"
00090 :
00091 "=r" (bit) /* 0, old bit value */
00092 :
00093 "m" (*dest), /* 1 mem, destination operand */
00094 "Ir" (b) /* 2, bit number */
00095 :
00096 "memory", "cc"
00097);
00098
00099 return (int)bit;
00100 }
00101
00102
00103 /* bit test and set */
00104 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00105 L4_INLINE int
00106 l4util_bts(int b, volatile l4_umword_t * dest)
00107 {
00108 l4_int8_t bit;
00109
00110 __asm__ __volatile__
00111 (
00112 "lock; bts %2,%1 \n\t"
00113 "setc %0 \n\t"
00114 :
00115 "=r" (bit) /* 0, old bit value */
00116 :
00117 "m" (*dest), /* 1 mem, destination operand */
00118 "Ir" (b) /* 2, bit number */
00119 :
00120 "memory", "cc"

```

```

00121);
00122
00123 return (int)bit;
00124 }
00125
00126 /* bit test and reset */
00127 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00128 L4_INLINE int
00129 l4util_btr(int b, volatile l4_umword_t * dest)
00130 {
00131 l4_int8_t bit;
00132
00133 __asm__ __volatile__
00134 (
00135 "lock; btr %2,%1 \n\t"
00136 "setc %0 \n\t"
00137 :
00138 "=r" (bit) /* 0, old bit value */
00139 :
00140 "m" (*dest), /* 1 mem, destination operand */
00141 "Ir" (b) /* 2, bit number */
00142 :
00143 "memory", "cc"
00144);
00145
00146 return (int)bit;
00147 }
00148
00149 /* bit test and complement */
00150 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00151 L4_INLINE int
00152 l4util_btc(int b, volatile l4_umword_t * dest)
00153 {
00154 l4_int8_t bit;
00155
00156 __asm__ __volatile__
00157 (
00158 "lock; btc %2,%1 \n\t"
00159 "setc %0 \n\t"
00160 :
00161 "=r" (bit) /* 0, old bit value */
00162 :
00163 "m" (*dest), /* 1 mem, destination operand */
00164 "Ir" ((l4_umword_t)b) /* 2, bit number */
00165 :
00166 "memory", "cc"
00167);
00168
00169 return (int)bit;
00170 }
00171
00172 /* bit scan reverse */
00173 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00174 L4_INLINE int
00175 l4util_bsr(l4_umword_t word)
00176 {
00177 l4_umword_t tmp;
00178
00179 if (L4_UNLIKELY(word == 0))
00180 return -1;
00181
00182 __asm__ __volatile__
00183 (
00184 "bsr %1,%0 \n\t"
00185 :
00186 "=r" (tmp) /* 0, index of most significant set bit */
00187 :
00188 "r" (word) /* 1, argument */
00189);
00190
00191 return tmp;
00192 }
00193
00194 /* bit scan forward */
00195 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00196 L4_INLINE int
00197 l4util_bsf(l4_umword_t word)
00198 {
00199 l4_umword_t tmp;
00200
00201 if (L4_UNLIKELY(word == 0))
00202 return -1;
00203
00204 __asm__ __volatile__
00205 (
00206 "bsf %1,%0 \n\t"
00207 :

```

```

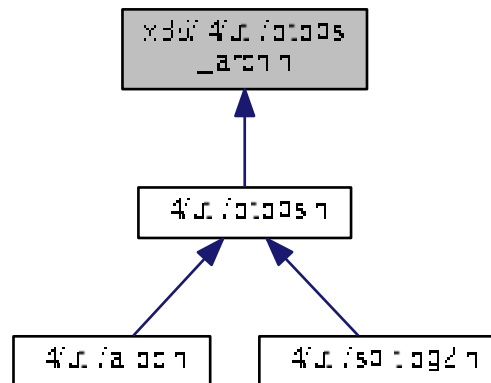
00208 "r" (tmp) /* 0, index of least significant set bit */
00209 :
00210 "r" (word) /* 1, argument */
00211);
00212
00213 return tmp;
00214 }
00215
00216 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00217 L4_INLINE int
00218 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00219 {
00220 l4_mword_t dummy0, dummy1, res;
00221
00222 __asm__ __volatile__
00223 (
00224 "xor %%rax,%%rax \n\t"
00225 "repe; scasl \n\t"
00226 "jz 1f \n\t"
00227 "lea -4(%%rdi),%%rdi \n\t"
00228 "bsfq (%%rdi),%%rax \n\t"
00229 "1: \n\t"
00230 "sub %%rbx,%%rdi \n\t"
00231 "shl $3,%%rdi \n\t"
00232 "add %%rdi,%%rax \n\t"
00233 :
00234 "a" (res), "=&c" (dummy0), "=&D" (dummy1)
00235 :
00236 "1" ((size + 31) >> 5), "2" (dest), "b" (dest)
00237 :
00238 "cc", "memory");
00239
00240 return res;
00241 }
00242
00243 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00244 L4_INLINE int
00245 l4util_find_first_zero_bit(const void * dest,
00246 l4_size_t size)
00247 {
00248 l4_mword_t dummy0, dummy1, dummy2, res;
00249
00250 if (!size)
00251 return 0;
00252
00253 __asm__ __volatile__
00254 (
00255 "mov $-1,%%rax \n\t"
00256 "xor %%rdx,%%rdx \n\t"
00257 "repe; scasl \n\t"
00258 "je 1f \n\t"
00259 "xor -4(%%rdi),%%rax \n\t"
00260 "sub $4,%%rdi \n\t"
00261 "bsf %%rax,%%rdx \n\t"
00262 "1: \n\t"
00263 "sub %[dest],%%rdi \n\t"
00264 "shl $3,%%rdi \n\t"
00265 "add %%rdi,%%rdx \n\t"
00266 :
00267 "d" (res), "=&c" (dummy0), "=&D" (dummy1), "=&a" (dummy2)
00268 :
00269 "1" ((size + 31) >> 5), "2" (dest), [dest] "S" (dest)
00270 :
00271 "cc", "memory");
00272
00273 return res;
00274 }
00275 EXTERN_C_END
00276
00277 #endif /* ! __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__ */

```

## 15.59 x86/I4/util/bitops\_arch.h File Reference

x86 bit manipulation functions

This graph shows which files directly or indirectly include this file:



### 15.59.1 Detailed Description

x86 bit manipulation functions

Date

07/03/2001

Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [bitops\\_arch.h](#).

## 15.60 bitops\_arch.h

```

00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #ifndef __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__
00019 #define __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__
00020
00021 /*****
00022 *** Implementation
00023 *****/
00024
00025 EXTERN_C_BEGIN
00026
00027 /* set bit */
00028 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT

```



```

00029 L4_INLINE void
00030 l4util_set_bit(int b, volatile l4_umword_t * dest)
00031 {
00032 __asm__ __volatile__
00033 (
00034 "lock; btsl %1,%0 \n\t"
00035 :
00036 :
00037 "m" (*dest), /* 0 mem, destination operand */
00038 "Ir" (b), /* 1, bit number */
00039 :
00040 "memory", "cc"
00041);
00042 }
00043
00044 /* clear bit */
00045 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00046 L4_INLINE void
00047 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00048 {
00049 __asm__ __volatile__
00050 (
00051 "lock; btrl %1,%0 \n\t"
00052 :
00053 :
00054 "m" (*dest), /* 0 mem, destination operand */
00055 "Ir" (b), /* 1, bit number */
00056 :
00057 "memory", "cc"
00058);
00059 }
00060
00061 /* change bit */
00062 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00063 L4_INLINE void
00064 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00065 {
00066 __asm__ __volatile__
00067 (
00068 "lock; btc1 %1,%0 \n\t"
00069 :
00070 :
00071 "m" (*dest), /* 0 mem, destination operand */
00072 "Ir" (b), /* 1, bit number */
00073 :
00074 "memory", "cc"
00075);
00076 }
00077
00078 /* test bit */
00079 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00080 L4_INLINE int
00081 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00082 {
00083 l4_int8_t bit;
00084
00085 __asm__ __volatile__
00086 (
00087 "btl %2,%1 \n\t"
00088 "setc %0 \n\t"
00089 :
00090 "=q" (bit) /* 0, old bit value */
00091 :
00092 "m" (*dest), /* 1 mem, destination operand */
00093 "Ir" (b), /* 2, bit number */
00094 :
00095 "memory", "cc"
00096);
00097
00098 return (int)bit;
00099 }
00100
00101 /* bit test and set */
00102 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00103 L4_INLINE int
00104 l4util_bts(int b, volatile l4_umword_t * dest)
00105 {
00106 l4_int8_t bit;
00107
00108 __asm__ __volatile__
00109 (
00110 "lock; btsl %2,%1 \n\t"
00111 "setc %0 \n\t"
00112 :
00113 "=q" (bit) /* 0, old bit value */
00114 :
00115 "m" (*dest), /* 1 mem, destination operand */

```

```

00116 "Ir" (b) /* 2, bit number */
00117 :
00118 "memory", "cc"
00119);
00120
00121 return (int)bit;
00122 }
00123
00124 /* bit test and reset */
00125 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00126 L4_INLINE int
00127 l4util_btr(int b, volatile l4_umword_t * dest)
00128 {
00129 l4_int8_t bit;
00130
00131 __asm__ __volatile__
00132 (
00133 "lock; btrl %2,%1 \n\t"
00134 "setc %0 \n\t"
00135 :
00136 "=q" (bit) /* 0, old bit value */
00137 :
00138 "m" (*dest), /* 1 mem, destination operand */
00139 "Ir" (b) /* 2, bit number */
00140 :
00141 "memory", "cc"
00142);
00143
00144 return (int)bit;
00145 }
00146
00147 /* bit test and complement */
00148 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00149 L4_INLINE int
00150 l4util_btc(int b, volatile l4_umword_t * dest)
00151 {
00152 l4_int8_t bit;
00153
00154 __asm__ __volatile__
00155 (
00156 "lock; btcl %2,%1 \n\t"
00157 "setc %0 \n\t"
00158 :
00159 "=q" (bit) /* 0, old bit value */
00160 :
00161 "m" (*dest), /* 1 mem, destination operand */
00162 "Ir" (b) /* 2, bit number */
00163 :
00164 "memory", "cc"
00165);
00166
00167 return (int)bit;
00168 }
00169
00170 /* bit scan reverse */
00171 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00172 L4_INLINE int
00173 l4util_bsr(l4_umword_t word)
00174 {
00175 int tmp;
00176
00177 if (L4_UNLIKELY(word == 0))
00178 return -1;
00179
00180 __asm__ __volatile__
00181 (
00182 "bsrl %1,%0 \n\t"
00183 :
00184 "=r" (tmp) /* 0, index of most significant set bit */
00185 :
00186 "r" (word) /* 1, argument */
00187);
00188
00189 return tmp;
00190 }
00191
00192 /* bit scan forward */
00193 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00194 L4_INLINE int
00195 l4util_bsf(l4_umword_t word)
00196 {
00197 int tmp;
00198
00199 if (L4_UNLIKELY(word == 0))
00200 return -1;
00201
00202 __asm__ __volatile__

```

```

00203 (
00204 "bsfl %1,%0 \n\t"
00205 :
00206 "=r" (tmp) /* 0, index of least significant set bit */
00207 :
00208 "r" (word) /* 1, argument */
00209);
00210
00211 return tmp;
00212 }
00213
00214 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00215 L4_INLINE int
00216 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00217 {
00218 l4_mword_t dummy0, dummy1, res;
00219
00220 __asm__ __volatile__
00221 (
00222 "repe; scasl \n\t"
00223 "jz 1f \n\t"
00224 "leal -4(%edi),%edi \n\t"
00225 "bsfl (%edi),%eax \n\t"
00226 "1: \n\t"
00227 "subl %%esi,%edi \n\t"
00228 "shll $3,%edi \n\t"
00229 "addl %edi,%eax \n\t"
00230 :
00231 "=a" (res), "=c" (dummy0), "=D" (dummy1)
00232 :
00233 "a" (0), "c" ((size+31) >> 5), "D" (dest), "S" (dest)
00234 :
00235 "cc", "memory");
00236
00237 return res;
00238 }
00239
00240 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00241 L4_INLINE int
00242 l4util_find_first_zero_bit(const void * dest,
00243 l4_size_t size)
00244 {
00245 l4_mword_t dummy0, dummy1, dummy2, res;
00246
00247 if (!size)
00248 return 0;
00249
00250 __asm__ __volatile__
00251 (
00252 "repe; scasl \n\t"
00253 "je 1f \n\t"
00254 "xorl -4(%edi),%eax \n\t"
00255 "subl $4,%edi \n\t"
00256 "bsfl %%eax,%%edx \n\t"
00257 "1: \n\t"
00258 "subl %%esi,%edi \n\t"
00259 "shll $3,%edi \n\t"
00260 "addl %edi,%%edx \n\t"
00261 :
00262 "=d" (res), "=c" (dummy0), "=D" (dummy1), "=a" (dummy2)
00263 :
00264 "a" (~0), "c" ((size+31) >> 5), "d" (0), "D" (dest), "S" (dest)
00265 :
00266 "cc", "memory");
00267
00268 return res;
00269 }
00270 EXTERN_C_END
00271
00272 #endif /* ! __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__ */

```

## 15.61 arm/l4/util/cpu.h File Reference

CPU related functions.

### 15.61.1 Detailed Description

CPU related functions.

**Author**

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

**15.62 cpu.h**

```

00001
00008 /*
00009 * (c) 2004-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef __L4_UTIL__ARCH_ARM__CPU_H__
00017 #define __L4_UTIL__ARCH_ARM__CPU_H__
00018
00019 /* Nothing yet */
00020
00021 #endif /* __L4_UTIL__ARCH_ARM__CPU_H__ */

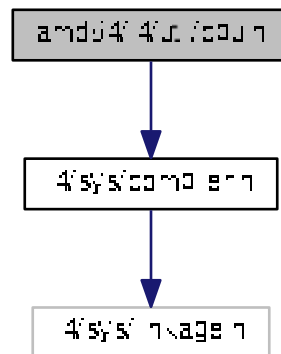
```

**15.63 amd64/l4/util/cpu.h File Reference**

CPU related functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cpu.h:

**Functions**

- `int l4util_cpu_has_cpuid (void)`  
Check whether the CPU supports the "cpuid" instruction.
- `unsigned int l4util_cpu_capabilities (void)`  
Returns the CPU capabilities if the "cpuid" instruction is available.
- `unsigned int l4util_cpu_capabilities_nocheck (void)`  
Returns the CPU capabilities.
- `void l4util_cpu_cpuid (unsigned long mode, unsigned long *eax, unsigned long *ebx, unsigned long *ecx, unsigned long *edx)`  
Generic CPUID access function.

### 15.63.1 Detailed Description

CPU related functions.

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 15.64 cpu.h

```

00001
00007 /*
00008 * (c) 2004-2009 Author(s)
00009 * economic rights: Technische Universität Dresden (Germany)
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014
00015 #ifndef __L4_UTIL_CPU_H
00016 #define __L4_UTIL_CPU_H
00017
00018 #include <14/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00027
00033 L4_INLINE int l4util_cpu_has_cpuid(void);
00034
00041 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00042
00048 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00049
00053 L4_INLINE void
00054 l4util_cpu_cpuid(unsigned long mode,
00055 unsigned long *eax, unsigned long *ebx,
00056 unsigned long *ecx, unsigned long *edx);
00057
00059 static inline void
00060 l4util_cpu_pause(void)
00061 {
00062 __asm__ __volatile__ ("rep; nop");
00063 }
00064
00065 L4_INLINE int
00066 l4util_cpu_has_cpuid(void)
00067 {
00068 return 1;
00069 }
00070
00071 L4_INLINE void
00072 l4util_cpu_cpuid(unsigned long mode,
00073 unsigned long *eax, unsigned long *ebx,
00074 unsigned long *ecx, unsigned long *edx)
00075 {
00076 asm volatile("cpuid"
00077 : "=a" (*eax),
00078 "=b" (*ebx),
00079 "=c" (*ecx),
00080 "=d" (*edx)
00081 : "a" (mode)
00082);
00083 }
00084
00085 L4_INLINE unsigned int
00086 l4util_cpu_capabilities_nocheck(void)
00087 {
00088 unsigned long dummy, capability;
00089
00090 /* get CPU capabilities */
00091 l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00092
00093 return capability;
00094 }

```

```

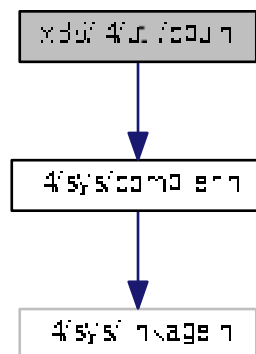
00095
00096 L4_INLINE unsigned int
00097 l4util_cpu_capabilities(void)
00098 {
00099 if (!l4util_cpu_has_cpuid())
00100 return 0; /* CPU has not cpuid instruction */
00101 return l4util_cpu_capabilities_nocheck();
00102 }
00103
00104 EXTERN_C_END
00105
00106 #endif
00107
00108

```

## 15.65 x86/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>
Include dependency graph for cpu.h:
```



## Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_nocheck](#) (void)  
*Returns the CPU capabilities.*
- void [l4util\\_cpu\\_cpuid](#) (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

## 15.65.1 Detailed Description

CPU related functions.

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 15.66 cpu.h

```

00001
00007 /*
00008 * (c) 2004-2009 Author(s)
00009 * economic rights: Technische Universität Dresden (Germany)
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014
00015 #ifndef __L4_UTIL_CPU_H
00016 #define __L4_UTIL_CPU_H
00017
00018 #include <l4/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00027
00033 L4_INLINE int l4util_cpu_has_cpuid(void);
00034
00041 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00042
00048 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00049
00053 L4_INLINE void
00054 l4util_cpu_cpuid(unsigned long mode,
00055 unsigned long *eax, unsigned long *ebx,
00056 unsigned long *ecx, unsigned long *edx);
00057
00059 static inline void
00060 l4util_cpu_pause(void)
00061 {
00062 __asm__ __volatile__ ("rep; nop");
00063 }
00064
00065 L4_INLINE int
00066 l4util_cpu_has_cpuid(void)
00067 {
00068 unsigned long eax;
00069
00070 asm volatile("pushl %%ebx \t\n"
00071 "pushfl \t\n"
00072 "popl %%eax \t\n" /* get eflags */
00073 "movl %%eax, %%ebx \t\n" /* save it */
00074 "xorl $0x200000, %%eax \t\n" /* toggle ID bit */
00075 "pushl %%eax \t\n"
00076 "popfl \t\n" /* set again */
00077 "pushfl \t\n"
00078 "popl %%eax \t\n" /* get it again */
00079 "xorl %%ebx, %%eax \t\n"
00080 "pushl %%ebx \t\n"
00081 "popfl \t\n" /* restore saved flags */
00082 "popl %%ebx \t\n"
00083 : "=a" (eax)
00084 : /* no input */
00085);
00086
00087 return eax & 0x200000;
00088 }
00089
00090 L4_INLINE void
00091 l4util_cpu_cpuid(unsigned long mode,
00092 unsigned long *eax, unsigned long *ebx,
00093 unsigned long *ecx, unsigned long *edx)
00094 {

```

```

00095 asm volatile("pushl %%ebx \t\n"
00096 "cpuid \t\n"
00097 "mov %%ebx, %%esi \t\n"
00098 "popl %%ebx \t\n"
00099 : "=a" (*eax),
00100 "=S" (*ebx),
00101 "=c" (*ecx),
00102 "=d" (*edx)
00103 : "a" (mode));
00104 }
00105
00106 L4_INLINE unsigned int
00107 l4util_cpu_capabilities_nocheck(void)
00108 {
00109 unsigned long dummy, capability;
00110
00111 /* get CPU capabilities */
00112 l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00113
00114 return capability;
00115 }
00116
00117 L4_INLINE unsigned int
00118 l4util_cpu_capabilities(void)
00119 {
00120 if (!l4util_cpu_has_cpuid())
00121 return 0; /* CPU has not cpuid instruction */
00122
00123 return l4util_cpu_capabilities_nocheck();
00124 }
00125
00126 EXTERN_C_END
00127
00128 #endif
00129

```

## 15.67 arm/l4/util/l4\_macros.h File Reference

Main function.

### 15.67.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.68 l4\_macros.h

```

00001
00008 /*
00009 * (c) 2006-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */

```



```

00015
00016 #ifndef _L4UTIL__ARCH_ARM__L4_MACROS_H
00017 #define _L4UTIL__ARCH_ARM__L4_MACROS_H
00018
00019 #include_next <l4/util/l4_macros.h>
00020
00021 #ifndef l4_addr_fmt
00022 # define l4_addr_fmt "%08lx"
00023 #endif
00024
00025 #endif /* !_L4UTIL__ARCH_ARM__L4_MACROS_H */

```

## 15.69 amd64/l4/util/l4\_macros.h File Reference

Main function.

### 15.69.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.70 l4\_macros.h

```

00001
00008 /*
00009 * (c) 2006-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef _L4UTIL__ARCH_AMD64__L4_MACROS_H
00017 #define _L4UTIL__ARCH_AMD64__L4_MACROS_H
00018
00019 #include_next <l4/util/l4_macros.h>
00020
00021 #ifndef l4_addr_fmt
00022 # define l4_addr_fmt "%016lx"
00023 #endif
00024
00025 #endif /* !_L4UTIL__ARCH_AMD64__L4_MACROS_H */

```

## 15.71 l4/util/l4\_macros.h File Reference

Some useful generic macros, L4f version.

### 15.71.1 Detailed Description

Some useful generic macros, L4f version.

#### Date

11/12/2002

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.72 l4\_macros.h

```

00001 /*****
00008 */
00009 * (c) 2000-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 /*****
00017
00018 #pragma once
00019
00020 /*****
00021 *** generic macros
00022 *****/
00023
00024 /* generate L4 thread id printf string */
00025 #ifndef l4util_idstr
00026 # define l4util_idfmt "%lx"
00027 # define l4util_idfmt_adjust "%04lx"
00028 # define l4util_idstr(tid) (tid >> L4_CAP_SHIFT)
00029 #endif
00030

```

## 15.73 x86/l4/uti/l4\_macros.h File Reference

Main function.

### 15.73.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.74 l4\_macros.h

```

00001
00008 /*
00009 * (c) 2006-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016
00017 #ifndef _L4UTIL__ARCH_X86__L4_MACROS_H
00018 #define _L4UTIL__ARCH_X86__L4_MACROS_H
00019
00020 #include_next <l4/util/l4_macros.h>
00021
00022 #ifndef l4_addr_fmt
00023 # define l4_addr_fmt "%08lx"
00024 #endif
00025
00026 #endif /* !_L4UTIL__ARCH_X86__L4_MACROS_H */

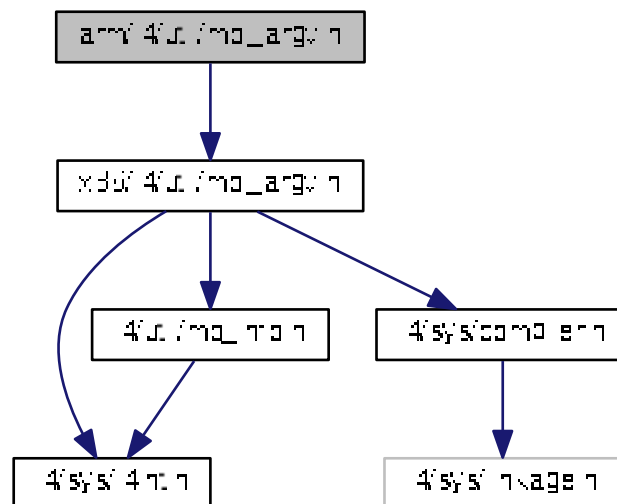
```

## 15.75 arm/l4/util/mbi\_argv.h File Reference

Multiboot.

```
#include <x86/l4/util/mbi_argv.h>
```

Include dependency graph for mbi\_argv.h:



### 15.75.1 Detailed Description

Multiboot.

Definition in file [mbi\\_argv.h](#).

## 15.76 mbi\_argv.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 /* If this persists, move it to a generic place */
00013 #include <x86/l4/util/mbi_argv.h>

```

## 15.77 amd64/l4/util/mbi\_argv.h File Reference

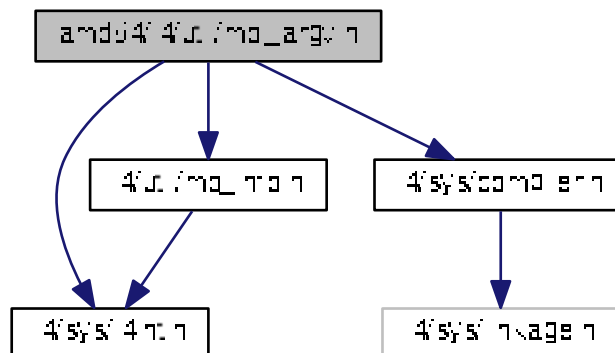
command line handling

```

#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for mbi\_argv.h:



### 15.77.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 15.78 mbi\_argv.h

```

00001
00008 /*
00009 * (c) 2003-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef L4UTIL_MBI_ARGV
00017 #define L4UTIL_MBI_ARGV
00018
00019 #include <l4/sys/l4int.h>
00020 #include <l4/util/mb_info.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_CV void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00026
00027 extern int l4util_argc;
00028 extern char *l4util_argv[];
00029
00030 EXTERN_C_END
00031
00032 #endif
00033

```

## 15.79 x86/l4/util/mbi\_argv.h File Reference

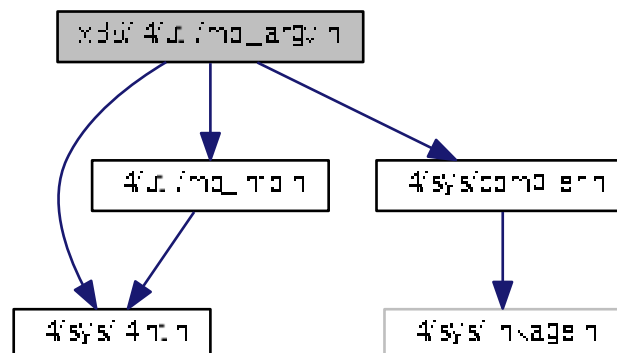
command line handling

```

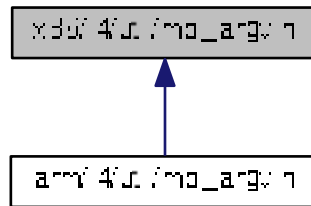
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for mbi\_argv.h:



This graph shows which files directly or indirectly include this file:



### 15.79.1 Detailed Description

command line handling

#### Date

2003

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 15.80 mbi\_argv.h

```

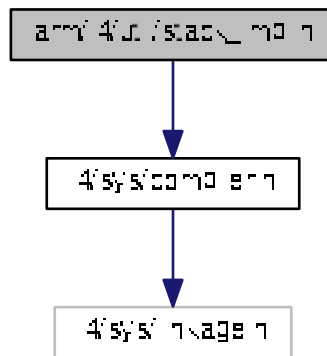
00001
00008 /*
00009 * (c) 2003-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef L4UTIL_MBI_ARGV
00017 #define L4UTIL_MBI_ARGV
00018
00019 #include <l4/sys/l4int.h>
00020 #include <l4/util/mb_info.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00026
00027 extern int l4util_argc;
00028 extern char *l4util_argv[];
00029
00030 EXTERN_C_END
00031
00032 #endif
00033

```

## 15.81 arm/l4/util/stack\_impl.h File Reference

Stack utilities, arm version.

```
#include <l4/sys/compiler.h>
Include dependency graph for stack_impl.h:
```



### Functions

- [l4\\_addr\\_t l4util\\_stack\\_get\\_sp](#) (void)  
*Get current stack pointer.*

#### 15.81.1 Detailed Description

Stack utilities, arm version.

Definition in file [stack\\_impl.h](#).

#### 15.81.2 Function Documentation

##### 15.81.2.1 l4util\_stack\_get\_sp()

```
l4_addr_t l4util_stack_get_sp (
 void) [inline]
```

Get current stack pointer.

##### Returns

stack pointer.

Definition at line 23 of file [stack\\_impl.h](#).

## 15.82 stack\_impl.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights; Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__
00013 #define __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__
00014
00015 #include <l4/sys/compiler.h>
00016
00017 EXTERN_C_BEGIN
00018
00019 #ifndef _L4UTIL_STACK_H
00020 #error Do not include stack_impl.h directly, use stack.h instead
00021 #endif
00022
00023 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00024 {
00025 register l4_addr_t sp asm ("sp");
00026 return sp;
00027 }
00028
00029 EXTERN_C_END
00030
00031 #endif /* ! __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__ */

```

## 15.83 amd64/l4/util/stack\_impl.h File Reference

Stack utilities for amd64.

### Functions

- [l4\\_addr\\_t l4util\\_stack\\_get\\_sp](#) (void)  
*Get current stack pointer.*

### 15.83.1 Detailed Description

Stack utilities for amd64.

Definition in file [stack\\_impl.h](#).

### 15.83.2 Function Documentation

#### 15.83.2.1 l4util\_stack\_get\_sp()

```

l4_addr_t l4util_stack_get_sp (
 void) [inline]

```

Get current stack pointer.

#### Returns

stack pointer.

Definition at line 22 of file [stack\\_impl.h](#).

References [EXTERN\\_C\\_END](#).



## 15.84 stack\_impl.h

```

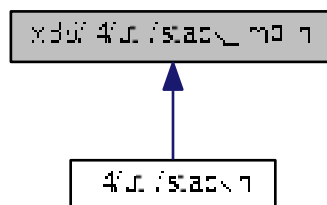
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__
00014 #define __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__
00015
00016 EXTERN_C_BEGIN
00017
00018 #ifndef __L4UTIL__STACK_H
00019 #error Do not include stack_impl.h directly, use stack.h instead
00020 #endif
00021
00022 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00023 {
00024 l4_addr_t rsp;
00025
00026 asm("movq %%rsp, %0\n\t" : "=r" (rsp) :);
00027 return rsp;
00028 }
00029
00030 EXTERN_C_END
00031
00032 #endif /* ! __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__ */

```

## 15.85 x86/l4/util/stack\_impl.h File Reference

Stack utilities for x86.

This graph shows which files directly or indirectly include this file:



### Functions

- [l4\\_addr\\_t l4util\\_stack\\_get\\_sp](#) (void)  
Get current stack pointer.

### 15.85.1 Detailed Description

Stack utilities for x86.

Definition in file [stack\\_impl.h](#).

## 15.85.2 Function Documentation

### 15.85.2.1 l4util\_stack\_get\_sp()

```
l4_addr_t l4util_stack_get_sp (
 void) [inline]
```

Get current stack pointer.

#### Returns

stack pointer.

Definition at line 21 of file [stack\\_impl.h](#).

References [EXTERN\\_C\\_END](#).

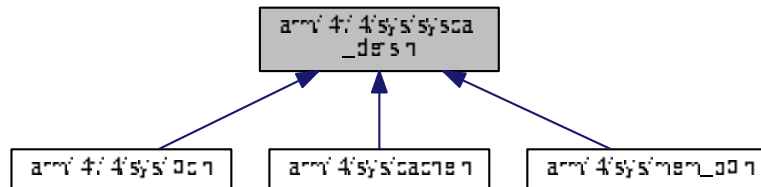
## 15.86 stack\_impl.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__INCLUDE__ARCH_X86__STACK_IMPL_H__
00013 #define __L4UTIL__INCLUDE__ARCH_X86__STACK_IMPL_H__
00014
00015 #ifndef _L4UTIL_STACK_H
00016 #error Do not include stack_impl.h directly, use stack.h instead
00017 #endif
00018
00019 EXTERN_C_BEGIN
00020
00021 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00022 {
00023 l4_addr_t esp;
00024
00025 asm("movl %%esp, %0\n\t" : "=r" (esp) :);
00026 return esp;
00027 }
00028
00029 EXTERN_C_END
00030
00031 #endif /* ! __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__ */
```

## 15.87 arm/l4f/l4/sys/syscall\_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:



### 15.87.1 Detailed Description

Syscall entry definitions.

Definition in file [syscall\\_defs.h](#).

## 15.88 syscall\_defs.h

```

00001
00005 /*
00006 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #ifndef __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00023 #define __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00024
00025 #ifndef L4_SYSCALL_MAGIC_OFFSET
00026 # define L4_SYSCALL_MAGIC_OFFSET 8
00027 #endif
00028 #define L4_SYSCALL_INVOKE (-0x00000004-L4_SYSCALL_MAGIC_OFFSET)
00029 #define L4_SYSCALL_MEM_OP (-0x00000008-L4_SYSCALL_MAGIC_OFFSET)
00030 #define L4_SYSCALL_DEBUGGER (-0x0000000C-L4_SYSCALL_MAGIC_OFFSET)
00031
00032 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */

```

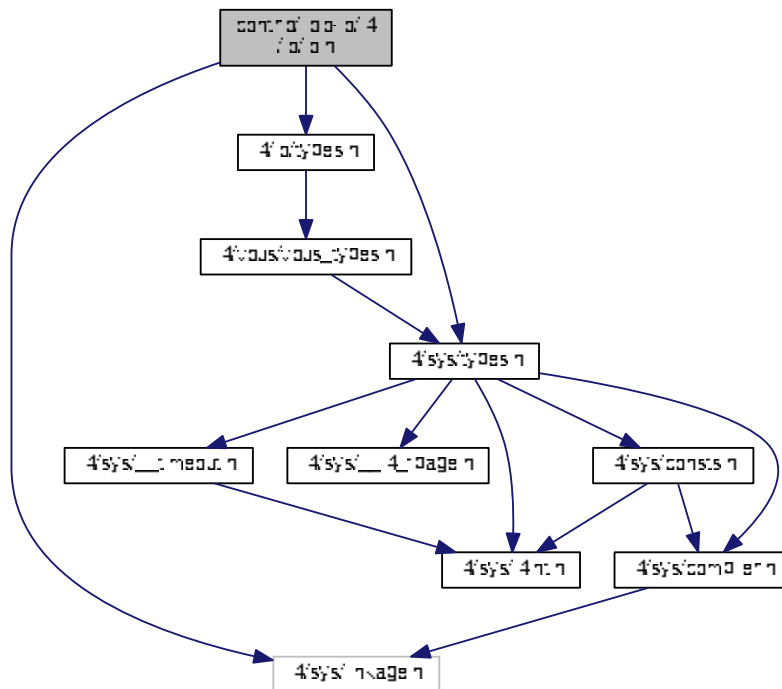
## 15.89 contrib/libio-io/l4/io/io.h File Reference

```

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

```

```
#include <l4/io/types.h>
Include dependency graph for io.h:
```



## Functions

- `l4_cap_idx_t l4io_request_icu` (void)  
*Request the ICU object of the client.*
- `long l4io_request_iomem` (`l4_addr_t` phys, unsigned long size, int flags, `l4_addr_t` \*virt)  
*Request an IO memory region.*
- `long l4io_request_iomem_region` (`l4_addr_t` phys, `l4_addr_t` virt, unsigned long size, int flags)  
*Request an IO memory region and map it to a specified region.*
- `long l4io_release_iomem` (`l4_addr_t` virt, unsigned long size)  
*Release an IO memory region.*
- `long l4io_search_iomem_region` (`l4_addr_t` phys, `l4_addr_t` size, `l4_addr_t` \*rstart, `l4_addr_t` \*rsize)  
*Search for a IO memory region.*
- `long l4io_request_ioport` (unsigned portnum, unsigned len)  
*Request an IO port region.*
- `long l4io_release_ioport` (unsigned portnum, unsigned len)  
*Release an IO port region.*
- `l4io_device_handle_t l4io_get_root_device` (void)  
*Get root device handle of the device bus.*
- `int l4io_iterate_devices` (`l4io_device_handle_t` \*devhandle, `l4io_device_t` \*dev, `l4io_resource_handle_t` \*reshandle)  
*Iterate over the device bus.*
- `int l4io_lookup_device` (const char \*devname, `l4io_device_handle_t` \*dev\_handle, `l4io_device_t` \*dev, `l4io_resource_handle_t` \*res\_handle)

*Find a device by name.*

- int [l4io\\_lookup\\_resource](#) (l4io\_device\_handle\_t devhandle, enum [l4io\\_resource\\_types\\_t](#) type, l4io\_resource\_handle\_t \*reshandle, [l4io\\_resource\\_t](#) \*res)

*Request a specific resource from a device description.*

- [l4\\_addr\\_t](#) [l4io\\_request\\_resource\\_iomem](#) (l4io\_device\_handle\_t devhandle, l4io\_resource\_handle\_t \*reshandle)

*Request IO memory.*

- void [l4io\\_request\\_all\\_ioports](#) (void(\*res\_cb)([l4vbus\\_resource\\_t](#) const \*res))

*Request all available IO-port resources.*

- int [l4io\\_has\\_resource](#) (enum [l4io\\_resource\\_types\\_t](#) type, l4vbus\_paddr\_t start, l4vbus\_paddr\_t end)

*Check if a resource is available.*

## 15.89.1 Function Documentation

### 15.89.1.1 l4io\_get\_root\_device()

```
l4io_device_handle_t l4io_get_root_device (
 void) [inline]
```

Get root device handle of the device bus.

#### Returns

root device handle

Definition at line 268 of file [io.h](#).

References [EXTERN\\_C\\_END](#).

### 15.89.1.2 l4io\_iterate\_devices()

```
int l4io_iterate_devices (
 l4io_device_handle_t * devhandle,
 l4io_device_t * dev,
 l4io_resource_handle_t * reshandle)
```

Iterate over the device bus.

#### Parameters

|                           |                                     |
|---------------------------|-------------------------------------|
| <a href="#">devhandle</a> | Device handle to start iterating at |
|---------------------------|-------------------------------------|

#### Return values

|                           |                    |
|---------------------------|--------------------|
| <a href="#">devhandle</a> | Next device handle |
|---------------------------|--------------------|

## Return values

|                  |                                 |
|------------------|---------------------------------|
| <i>dev</i>       | Device information, may be NULL |
| <i>reshandle</i> | Resource handle.                |

## Returns

0 on success, error code otherwise

## 15.89.1.3 l4io\_request\_all\_ioports()

```
void l4io_request_all_ioports (
 void(*) (l4vbus_resource_t const *res) res_cb)
```

Request all available IO-port resources.

## Parameters

|               |                                                                                    |
|---------------|------------------------------------------------------------------------------------|
| <i>res_cb</i> | Callback function called for every port resource found, give NULL for no callback. |
|---------------|------------------------------------------------------------------------------------|

## 15.89.1.4 l4io\_request\_icu()

```
l4_cap_idx_t l4io_request_icu (
 void)
```

Request the ICU object of the client.

## Returns

Client ICU object, an invalid capability selector is returned if no ICU is available.

## 15.90 io.h

```
00001
00004 /*
00005 * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00006 * Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012
00013
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/linkage.h>
00018 #include <l4/io/types.h>
00019
```

```

00020 EXTERN_C_BEGIN
00021
00038 L4_CV long L4_EXPORT
00039 l4io_request_irq(int irqnum, l4_cap_idx_t irqcap);
00040
00047 L4_CV l4_cap_idx_t L4_EXPORT
00048 l4io_request_icu(void);
00049
00061 L4_CV long L4_EXPORT
00062 l4io_release_irq(int irqnum, l4_cap_idx_t irq_cap);
00063
00088 L4_CV long L4_EXPORT
00089 l4io_request_iomem(l4_addr_t phys, unsigned long size, int flags,
00090 l4_addr_t *virt);
00091
00113 L4_CV long L4_EXPORT
00114 l4io_request_iomem_region(l4_addr_t phys,
00115 l4_addr_t virt,
00116 unsigned long size, int flags);
00116
00124 L4_CV long L4_EXPORT
00125 l4io_release_iomem(l4_addr_t virt, unsigned long size);
00126
00136 L4_CV long L4_EXPORT
00137 l4io_search_iomem_region(l4_addr_t phys,
00138 l4_addr_t size,
00139 l4_addr_t *rstart, l4_addr_t *rsize);
00139
00149 L4_CV long L4_EXPORT
00150 l4io_request_ioport(unsigned portnum, unsigned len);
00151
00161 L4_CV long L4_EXPORT
00162 l4io_release_ioport(unsigned portnum, unsigned len);
00163
00164
00165 /* ----- Device handling ----- */
00166
00172 L4_INLINE
00173 l4io_device_handle_t l4io_get_root_device(void);
00174
00184 L4_CV int L4_EXPORT
00185 l4io_iterate_devices(l4io_device_handle_t *devhandle,
00186 l4io_device_t *dev, l4io_resource_handle_t *reshandle);
00187
00199 L4_CV int L4_EXPORT
00200 l4io_lookup_device(const char *devname,
00201 l4io_device_handle_t *dev_handle,
00202 l4io_device_t *dev, l4io_resource_handle_t *res_handle);
00203
00218 L4_CV int L4_EXPORT
00219 l4io_lookup_resource(l4io_device_handle_t devhandle,
00220 enum l4io_resource_types_t type,
00221 l4io_resource_handle_t *reshandle,
00222 l4io_resource_t *res);
00223
00224
00225 /* ----- Convenience functions ----- */
00226
00239 L4_CV l4_addr_t L4_EXPORT
00240 l4io_request_resource_iomem(l4io_device_handle_t devhandle,
00241 l4io_resource_handle_t *reshandle);
00242
00249 L4_CV void L4_EXPORT
00250 l4io_request_all_ioports(void (*res_cb)(
00251 l4vbus_resource_t const *res));
00251
00260 L4_CV int L4_EXPORT
00261 l4io_has_resource(enum l4io_resource_types_t type,
00262 l4vbus_paddr_t start, l4vbus_paddr_t end);
00263
00264 /* ----- Implementations ----- */
00265 /* Implementations */
00266
00267 L4_INLINE
00268 l4io_device_handle_t l4io_get_root_device(void)
00269 { return 0; }
00270
00271 EXTERN_C_END

```

## 15.91 l4/sys/types.h File Reference

Common L4 ABI Data Types.





```

L4_PROTO_SIGMA0 = -6L, L4_PROTO_IO_PAGE_FAULT = -8L, L4_PROTO_KOBJECT = -10L, L4_PROTO_TASK = -11L,
L4_PROTO_THREAD = -12L, L4_PROTO_LOG = -13L, L4_PROTO_SCHEDULER = -14L, L4_PROTO_FACTORY = -15L,
L4_PROTO_VM = -16L, L4_PROTO_DMA_SPACE = -17L, L4_PROTO_IRQ_SENDER = -18L, L4_PROTO_IRQ_MUX = -19L,
L4_PROTO_SEMAPHORE = -20L, L4_PROTO_META = -21L, L4_PROTO_IOMMU = -22L, L4_PROTO_DEBUGGER = -23L,
L4_PROTO_SMCCC = -24L }

```

*Message tag for IPC operations.*

- enum `l4_msgtag_flags` {  
`L4_MSGTAG_ERROR`, `L4_MSGTAG_TRANSFER_FPU`, `L4_MSGTAG_SCHEDULE`, `L4_MSGTAG_PRIVILEGE`,  
`L4_MSGTAG_FLAGS` }

*Flags for message tags.*

## Functions

- `l4_msgtag_t l4_msgtag` (long label, unsigned words, unsigned items, unsigned flags) `L4_NOTHROW`  
*Create a message tag from the specified values.*
- long `l4_msgtag_label` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the protocol of tag.*
- unsigned `l4_msgtag_words` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the number of untyped words.*
- unsigned `l4_msgtag_items` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the number of typed items.*
- unsigned `l4_msgtag_flags` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the flags.*
- unsigned `l4_msgtag_has_error` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for error indicator flag.*
- unsigned `l4_msgtag_is_page_fault` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for page-fault protocol.*
- unsigned `l4_msgtag_is_preemption` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for preemption protocol.*
- unsigned `l4_msgtag_is_sys_exception` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for system-exception protocol.*
- unsigned `l4_msgtag_is_exception` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for exception protocol.*
- unsigned `l4_msgtag_is_sigma0` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for sigma0 protocol.*
- unsigned `l4_msgtag_is_io_page_fault` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for IO-page-fault protocol.*
- unsigned `l4_is_invalid_cap` (`l4_cap_idx_t` c) `L4_NOTHROW`  
*Test if a capability selector is the invalid capability.*
- unsigned `l4_is_valid_cap` (`l4_cap_idx_t` c) `L4_NOTHROW`  
*Test if a capability selector is a valid selector.*
- unsigned `l4_capability_equal` (`l4_cap_idx_t` c1, `l4_cap_idx_t` c2) `L4_NOTHROW`  
*Test if two capability selectors are equal.*
- `l4_cap_idx_t l4_capability_next` (`l4_cap_idx_t` c) `L4_NOTHROW`  
*Get the next capability selector after c.*

### 15.91.1 Detailed Description

Common [L4](#) ABI Data Types.

Definition in file [types.h](#).

### 15.91.2 Function Documentation

#### 15.91.2.1 l4\_capability\_next()

```
l4_cap_idx_t l4_capability_next (
 l4_cap_idx_t c) [inline]
```

Get the next capability selector after `c`.

#### Parameters

|                |                                                                        |
|----------------|------------------------------------------------------------------------|
| <code>c</code> | The capability selector for which the next selector shall be computed. |
|----------------|------------------------------------------------------------------------|

#### Returns

The next capability selector after `c`.

Definition at line [460](#) of file [types.h](#).

## 15.92 types.h

```
00001 /*****/
00002 /*
00003 * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00006 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 /*****/
00023 #pragma once
00024
00025 #include <l4/sys/l4int.h>
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/consts.h>
00028
00029 enum l4_msgtag_protocol
```

```

00050 {
00051 L4_PROTO_NONE = 0,
00052 L4_PROTO_ALLOW_SYSCALL = 1,
00053 L4_PROTO_PF_EXCEPTION = 1,
00054
00055 L4_PROTO_IRQ = -1L,
00056 L4_PROTO_PAGE_FAULT = -2L,
00057 L4_PROTO_PREEMPTION = -3L,
00058 L4_PROTO_SYS_EXCEPTION = -4L,
00059 L4_PROTO_EXCEPTION = -5L,
00060 L4_PROTO_SIGMA0 = -6L,
00061 L4_PROTO_IO_PAGE_FAULT = -8L,
00062 L4_PROTO_KOBJECT = -10L,
00063 L4_PROTO_TASK = -11L,
00064 L4_PROTO_THREAD = -12L,
00065 L4_PROTO_LOG = -13L,
00066 L4_PROTO_SCHEDULER = -14L,
00067 L4_PROTO_FACTORY = -15L,
00068 L4_PROTO_VM = -16L,
00069 L4_PROTO_DMA_SPACE = -17L,
00070 L4_PROTO_IRQ_SENDER = -18L,
00071 L4_PROTO_IRQ_MUX = -19L,
00072 L4_PROTO_SEMAPHORE = -20L,
00073 L4_PROTO_META = -21L,
00074 L4_PROTO_IOMMU = -22L,
00075 L4_PROTO_DEBUGGER = -23L,
00076 L4_PROTO_SMCCC = -24L,
00077 };
00078
00079 enum L4_varg_type
00080 {
00081 L4_VARG_TYPE_NIL = 0x00,
00082 L4_VARG_TYPE_UMWORD = 0x01,
00083 L4_VARG_TYPE_MWORD = 0x81,
00084 L4_VARG_TYPE_STRING = 0x02,
00085 L4_VARG_TYPE_FPAGE = 0x03,
00086
00087 L4_VARG_TYPE_SIGN = 0x80,
00088 };
00089
00090
00095 enum l4_msgtag_flags
00096 {
00097 // flags for received IPC
00102 L4_MSGTAG_ERROR = 0x8000,
00103
00104 // flags for sending IPC
00114 L4_MSGTAG_TRANSFER_FPU = 0x1000,
00123 L4_MSGTAG_SCHEDULE = 0x2000,
00133 L4_MSGTAG_PROPAGATE = 0x4000,
00134
00139 L4_MSGTAG_FLAGS = 0xf000,
00140 };
00141
00142
00159 typedef struct l4_msgtag_t
00160 {
00161 l4_mword_t raw;
00162 #ifdef __cplusplus
00163 long label() const throw() { return raw >> 16; }
00166 void label(long v) throw() { raw = (raw & 0xffff) | ((l4_umword_t)v << 16); }
00168 unsigned words() const throw() { return raw & 0x3f; }
00170 unsigned items() const throw() { return (raw >> 6) & 0x3f; }
00177 unsigned flags() const throw() { return raw & 0xf000; }
00179 bool is_page_fault() const throw() { return label() ==
L4_PROTO_PAGE_FAULT; }
00181 bool is_preemption() const throw() { return label() ==
L4_PROTO_PREEMPTION; }
00183 bool is_sys_exception() const throw() { return label() ==
L4_PROTO_SYS_EXCEPTION; }
00185 bool is_exception() const throw() { return label() ==
L4_PROTO_EXCEPTION; }
00187 bool is_sigma0() const throw() { return label() ==
L4_PROTO_SIGMA0; }
00189 bool is_io_page_fault() const throw() { return label() ==
L4_PROTO_IO_PAGE_FAULT; }
00191 unsigned has_error() const throw() { return raw & L4_MSGTAG_ERROR; }
00192 #endif
00193 } l4_msgtag_t;
00194
00195
00196
00208 L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned
words, unsigned items,
00209 unsigned flags) L4_NOTHROW;
00210
00219 L4_INLINE long l4_msgtag_label(l4_msgtag_t t)

```

```

 L4_NOTHROW;
00220
00229 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t)
 L4_NOTHROW;
00230
00239 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t)
 L4_NOTHROW;
00240
00251 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t)
 L4_NOTHROW;
00252
00265 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t)
 L4_NOTHROW;
00266
00275 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t)
 L4_NOTHROW;
00276
00284 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t)
 L4_NOTHROW;
00285
00294 L4_INLINE unsigned l4_msgtag_is_sys_exception(
 l4_msgtag_t t) L4_NOTHROW;
00295
00304 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t)
 L4_NOTHROW;
00305
00314 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t)
 L4_NOTHROW;
00315
00324 L4_INLINE unsigned l4_msgtag_is_io_page_fault(
 l4_msgtag_t t) L4_NOTHROW;
00325
00342 typedef unsigned long l4_cap_idx_t;
00343
00353 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00354
00364 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00365
00376 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2)
 L4_NOTHROW;
00377
00386 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c)
 L4_NOTHROW;
00387
00388 /* ***** */
00389 /* Implementation */
00390
00391 L4_INLINE unsigned
00392 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00393 { return c & L4_INVALID_CAP_BIT; }
00394
00395 L4_INLINE unsigned
00396 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00397 { return !(c & L4_INVALID_CAP_BIT); }
00398
00399 L4_INLINE unsigned
00400 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00401 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00402
00403
00407 L4_INLINE
00408 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00409 unsigned flags) L4_NOTHROW
00410 {
00411 return (l4_msgtag_t){ (l4_mword_t)((l4_umword_t)label << 16)
00412 | (l4_mword_t)(words & 0x3f)
00413 | (l4_mword_t)((items & 0x3f) << 6)
00414 | (l4_mword_t)(flags & 0xf000)};
00415 }
00416
00417
00418
00419 L4_INLINE
00420 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00421 { return t.raw >> 16; }
00422
00423 L4_INLINE
00424 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00425 { return t.raw & 0x3f; }
00426
00427 L4_INLINE
00428 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00429 { return (t.raw >> 6) & 0x3f; }
00430
00431 L4_INLINE
00432 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00433 { return t.raw & 0xf000; }

```

```

00434
00435
00436 L4_INLINE
00437 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00438 { return t.raw & L4_MSGTAG_ERROR; }
00439
00440
00441
00442 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00443 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00444
00445 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW
00446 { return l4_msgtag_label(t) == L4_PROTO_PREEMPTION; }
00447
00448 L4_INLINE unsigned l4_msgtag_is_sys_exception(
00449 l4_msgtag_t t) L4_NOTHROW
00449 { return l4_msgtag_label(t) == L4_PROTO_SYS_EXCEPTION; }
00450
00451 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00452 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }
00453
00454 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00455 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00456
00457 L4_INLINE unsigned l4_msgtag_is_io_page_fault(
00458 l4_msgtag_t t) L4_NOTHROW
00458 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00459
00460 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00461 { return c + L4_CAP_OFFSET; }
00462
00463 #include <l4/sys/__l4_fpage.h>
00464 #include <l4/sys/__timeout.h>

```

## 15.93 l4/cxx/avl\_map File Reference

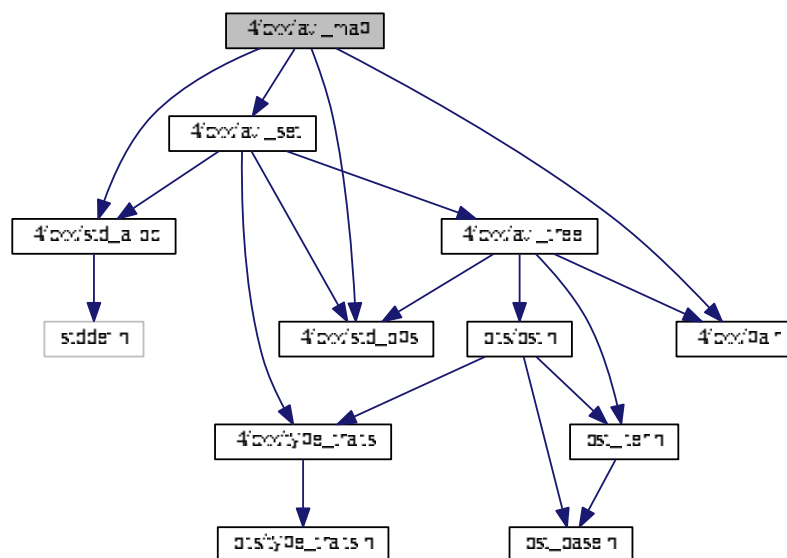
AVL map.

```

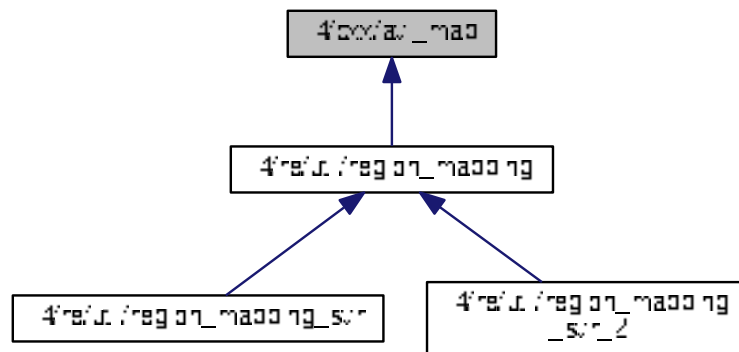
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
#include <l4/cxx/avl_set>

```

Include dependency graph for avl\_map:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [cxx::Bits::Avl\\_map\\_get\\_key< KEY\\_TYPE >](#)  
*Key-getter for [Avl\\_map](#).*
- class [cxx::Avl\\_map< KEY\\_TYPE, DATA\\_TYPE, COMPARE, ALLOC >](#)  
*AVL tree based associative container.*

## Namespaces

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the [cxx](#) package.*

### 15.93.1 Detailed Description

AVL map.

Definition in file [avl\\_map](#).

## 15.94 avl\_map

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate

```

```

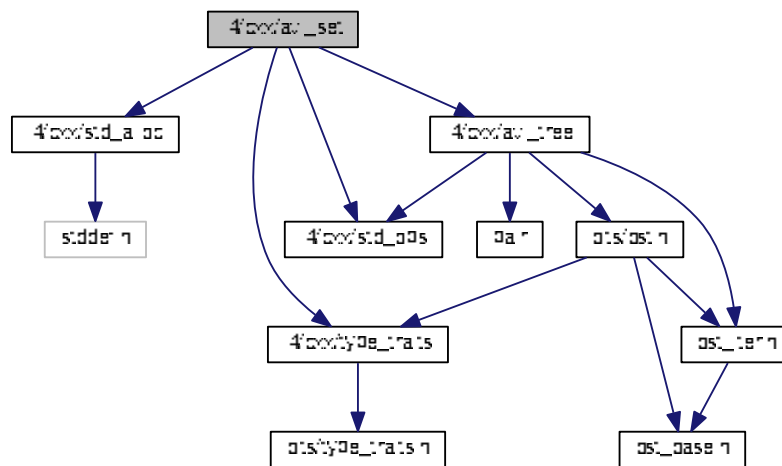
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023
00024 #pragma once
00025
00026 #include <l4/cxx/std_alloc>
00027 #include <l4/cxx/std_ops>
00028 #include <l4/cxx/pair>
00029 #include <l4/cxx/avl_set>
00030
00031 namespace cxx {
00032 namespace Bits {
00033
00035 template<typename KEY_TYPE>
00036 struct Avl_map_get_key
00037 {
00038 typedef KEY_TYPE Key_type;
00039 template<typename NODE>
00040 static Key_type const &key_of(NODE const *n)
00041 { return n->item.first; }
00042 };
00043 }
00044
00053 template< typename KEY_TYPE, typename DATA_TYPE,
00054 template<typename A> class COMPARE = Lt_functor,
00055 template<typename B> class ALLOC = New_allocator >
00056 class Avl_map :
00057 public Bits::Base_avl_set<Pair<KEY_TYPE, DATA_TYPE>,
00058 COMPARE<KEY_TYPE>, ALLOC,
00059 Bits::Avl_map_get_key<KEY_TYPE> >
00060 {
00061 private:
00062 typedef Pair<KEY_TYPE, DATA_TYPE> Local_item_type;
00063 typedef Bits::Base_avl_set<Local_item_type, COMPARE<KEY_TYPE>
00064 , ALLOC,
00065 Bits::Avl_map_get_key<KEY_TYPE> >
00066 Base_type;
00067
00068 public:
00069 typedef COMPARE<KEY_TYPE> Key_compare;
00070 typedef KEY_TYPE Key_type;
00071 typedef DATA_TYPE Data_type;
00072 typedef typename Base_type::Node Node;
00073 typedef typename Base_type::Node_allocator
00074 Node_allocator;
00075
00076 typedef typename Base_type::Iterator Iterator;
00077 typedef typename Base_type::Iterator iterator;
00078 typedef typename Base_type::Const_iterator Const_iterator;
00079 typedef typename Base_type::Const_iterator const_iterator;
00080 typedef typename Base_type::Rev_iterator Rev_iterator;
00081 typedef typename Base_type::Rev_iterator reverse_iterator;
00082 typedef typename Base_type::Const_rev_iterator Const_rev_iterator;
00083 typedef typename Base_type::Const_rev_iterator const_reverse_iterator;
00084
00085 Avl_map(Node_allocator const &alloc = Node_allocator())
00086 : Base_type(alloc)
00087 {}
00088
00089 cxx::Pair<Iterator, int> insert(Key_type const &key, Data_type const &data)
00090 { return Base_type::insert(Pair<Key_type, Data_type>(key, data)); }
00091
00092 Data_type const &operator [] (Key_type const &key) const
00093 { return this->find_node(key)->second; }
00094
00095 Data_type &operator [] (Key_type const &key)
00096 {
00097 Node n = this->find_node(key);
00098 if (n)
00099 return const_cast<Data_type&>(n->second);
00100 else
00101 return insert(key, Data_type()).first->second;
00102 }
00103 };
00104 }
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141

```

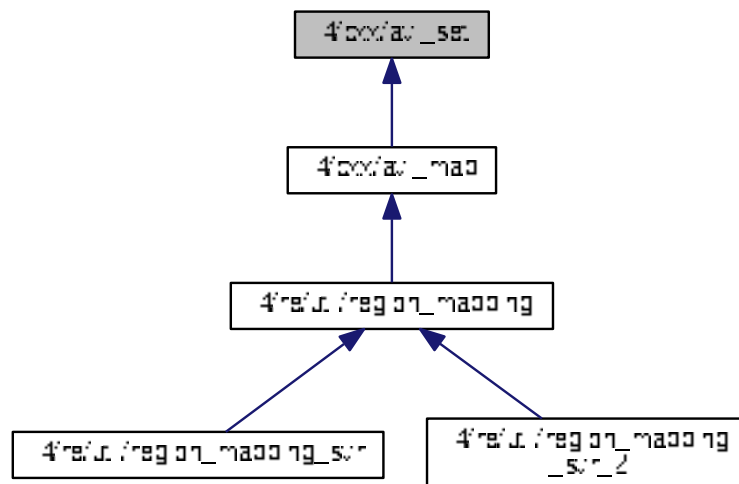
## 15.95 l4/cxx/avl\_set File Reference

AVL set.

```
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/type_traits>
#include <l4/cxx/avl_tree>
Include dependency graph for avl_set:
```



This graph shows which files directly or indirectly include this file:





## Data Structures

- `struct cxx::Bits::Avl_set_get_key< KEY_TYPE >`  
*Internal, key-getter for `Avl_set` nodes.*
- `class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`  
*Internal: AVL set with internally managed nodes.*
- `class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node`  
*A smart pointer to a tree item.*
- `class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >`  
*AVL set for simple compareable items.*

## Namespaces

- `cxx`  
*Our C++ library.*
- `cxx::Bits`  
*Internal helpers for the `cxx` package.*

### 15.95.1 Detailed Description

AVL set.

Definition in file `avl_set`.

## 15.96 avl\_set

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/cxx/std_alloc>
00028 #include <l4/cxx/std_ops>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/cxx/avl_tree>
00031
00032 namespace cxx {
00033 namespace Bits {
00042 template< typename Node, typename Key, typename Node_op >
00043 class Avl_set_iter : public __Bst_iter_b<Node, Node_op>
00044 {
00045 private:
00047 typedef __Bst_iter_b<Node, Node_op> Base;
00048
00049 using Base::_n;
00050 using Base::_r;

```

```

00051 using Base::inc;
00052
00053 public:
00054 Avl_set_iter() {}
00055
00062 Avl_set_iter(Node const *t) : Base(t) {}
00063
00064 Avl_set_iter(Base const &o) : Base(o) {}
00065
00066 Avl_set_iter(Avl_set_iter<Node, typename Type_traits<Key>::Non_const_type, Node_op> const &o)
00067 : Base(o) {}
00068
00073 Key &operator * () const { return const_cast<Node*>(_n)->item; }
00078 Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
00082 Avl_set_iter &operator ++ () { inc(); return *this; }
00086 Avl_set_iter &operator ++ (int)
00087 { Avl_set_iter tmp = *this; inc(); return tmp; }
00088
00089 };
00090
00092 template<typename KEY_TYPE>
00093 struct Avl_set_get_key
00094 {
00095 typedef KEY_TYPE Key_type;
00096 template<typename NODE>
00097 static Key_type const &key_of(NODE const *n)
00098 { return n->item; }
00099 };
00100
00101
00114 template< typename ITEM_TYPE, class COMPARE,
00115 template<typename A> class ALLOC,
00116 typename GET_KEY>
00117 class Base_avl_set
00118 {
00119 public:
00126 enum
00127 {
00128 E_noent = 2,
00129 E_exist = 17,
00130 E_nomem = 12,
00131 E_inval = 22
00132 };
00134 typedef ITEM_TYPE Item_type;
00136 typedef GET_KEY Get_key;
00138 typedef typename GET_KEY::Key_type Key_type;
00140 typedef typename Type_traits<Item_type>::Const_type Const_item_type;
00142 typedef COMPARE Item_compare;
00143
00144 private:
00146 class _Node : public Avl_tree_node
00147 {
00148 public:
00150 Item_type item;
00151
00152 _Node() : Avl_tree_node(), item() {}
00153
00154 _Node(Item_type const &item) : Avl_tree_node(), item(item) {}
00155 };
00156
00157 public:
00161 class Node
00162 {
00163 private:
00164 struct No_type;
00165 friend class Base_avl_set<ITEM_TYPE, COMPARE, ALLOC, GET_KEY>;
00166 _Node const *_n;
00167 explicit Node(_Node const *n) : _n(n) {}
00168
00169 public:
00171 Node() : _n(0) {}
00172
00174 Node &operator = (Node const &o) { _n = o._n; return *this; }
00175
00181 Item_type const &operator * () { return _n->item; }
00187 Item_type const *operator -> () { return &_n->item; }
00188
00193 bool valid() const { return _n; }
00194
00196 operator Item_type const * () { if (_n) return &_n->item; else return 0; }
00197 };
00198
00200 typedef ALLOC<_Node> Node_allocator;
00201
00202 private:
00203 typedef Avl_tree<_Node, GET_KEY, COMPARE>
 Tree;

```

```

00204 Tree _tree;
00206 Node_allocator _alloc;
00207
00208 Base_avl_set &operator = (Base_avl_set const &) = delete;
00209
00210 typedef typename Tree::Fwd_iter_ops Fwd;
00211 typedef typename Tree::Rev_iter_ops Rev;
00212
00213 public:
00214 typedef typename Type_traits<Item_type>::Param_type Item_param_type;
00215
00217 typedef Avl_set_iter<Node, Item_type, Fwd> Iterator;
00218 typedef Iterator iterator;
00220 typedef Avl_set_iter<Node, Const_item_type, Fwd> Const_iterator;
00221 typedef Const_iterator const_iterator;
00223 typedef Avl_set_iter<Node, Item_type, Rev> Rev_iterator;
00224 typedef Rev_iterator reverse_iterator;
00226 typedef Avl_set_iter<Node, Const_item_type, Rev> Const_rev_iterator;
00227 typedef Const_rev_iterator const_reverse_iterator;
00228
00235 explicit Base_avl_set(Node_allocator const &alloc = Node_allocator())
00236 : _tree(), _alloc(alloc)
00237 {}
00238
00239 ~Base_avl_set()
00240 {
00241 _tree.remove_all([this](_Node *n)
00242 {
00243 n->~_Node();
00244 _alloc.free(n);
00245 });
00246 }
00247
00255 inline Base_avl_set(Base_avl_set const &o);
00256
00274 cxx::Pair<Iterator, int> insert(Item_type const &item);
00275
00285 int remove(Key_type const &item)
00286 {
00287 _Node *n = _tree.remove(item);
00288 if ((long)n == -E_inval)
00289 return -E_inval;
00290
00291 if (n)
00292 {
00293 n->~_Node();
00294 _alloc.free(n);
00295 return 0;
00296 }
00297
00298 return -E_noent;
00299 }
00300
00305 int erase(Key_type const &item)
00306 { return remove(item); }
00307
00316 Node find_node(Key_type const &item) const
00317 { return Node(_tree.find_node(item)); }
00318
00327 Node lower_bound_node(Key_type const &key) const
00328 { return Node(_tree.lower_bound_node(key)); }
00329
00330 Node lower_bound_node(Key_type &&key) const
00331 { return Node(_tree.lower_bound_node(key)); }
00332
00337 Const_iterator begin() const { return _tree.begin(); }
00342 Const_iterator end() const { return _tree.end(); }
00343
00348 Iterator begin() { return _tree.begin(); }
00353 Iterator end() { return _tree.end(); }
00354
00359 Const_rev_iterator rbegin() const { return _tree.rbegin(); }
00364 Const_rev_iterator rend() const { return _tree.rend(); }
00365
00370 Rev_iterator rbegin() { return _tree.rbegin(); }
00375 Rev_iterator rend() { return _tree.rend(); }
00376
00377 Const_iterator find(Key_type const &item) const
00378 { return _tree.find(item); }
00379 };
00380
00381
00382 //-----
00383 /* Implementation of AVL Tree */
00384
00385 /* Create a copy */
00386 template< typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>

```

```

00387 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Base_avl_set
 (Base_avl_set const &o)
00388 : _tree(), _alloc(o._alloc)
00389 {
00390 for (Const_iterator i = o.begin(); i != o.end(); ++i)
00391 insert(*i);
00392 }
00393
00394 /* Insert new _Node. */
00395 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00396 Pair<typename Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Iterator
 , int>
00397 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::insert(Item
 const &item)
00398 {
00399 _Node *n = _alloc.alloc();
00400 if (!n)
00401 return cxx::pair(end(), -E_nomem);
00402
00403 new (n, Nothrow()) _Node(item);
00404 Pair<_Node *, bool> err = _tree.insert(n);
00405 if (!err.second)
00406 {
00407 n->~_Node();
00408 _alloc.free(n);
00409 }
00410
00411 return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.
 first)), err.second ? 0 : -E_exist);
00412 }
00413
00414 } // namespace Bits
00415
00427 template< typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>,
00428 template<typename A> class ALLOC = New_allocator>
00429 class Avl_set :
00430 public Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00431 Bits::Avl_set_get_key<ITEM_TYPE> >
00432 {
00433 private:
00434 typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00435 Bits::Avl_set_get_key<ITEM_TYPE> >
00436 Base;
00437 public:
00438 typedef typename Base::Node_allocator Node_allocator;
00439 Avl_set() = default;
00440 Avl_set(Node_allocator const &alloc)
00441 : Base(alloc)
00442 {}
00443 };
00444
00445 } // namespace cxx

```

## 15.97 l4/cxx/avl\_tree File Reference

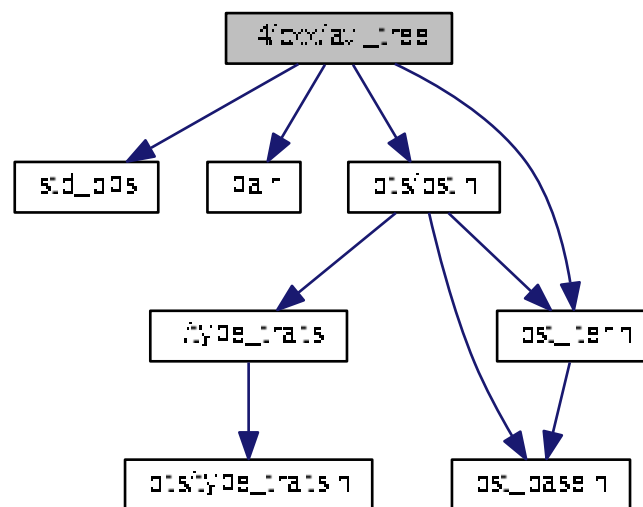
AVL tree.

```

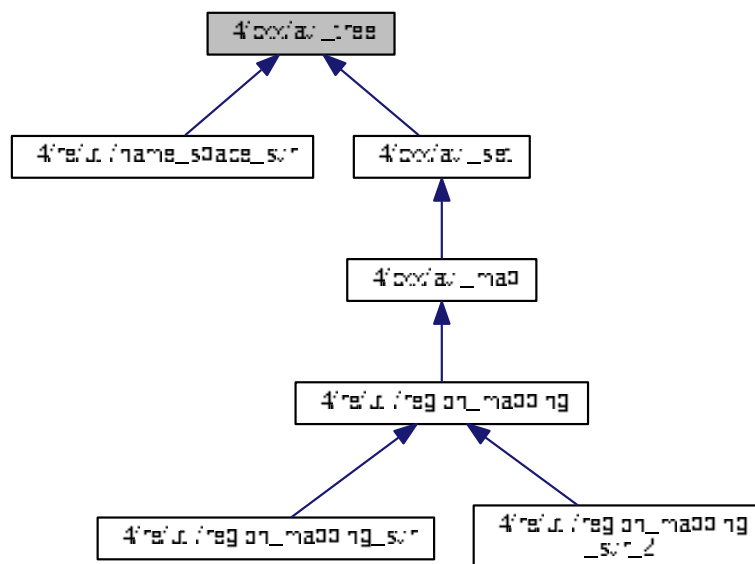
#include "std_ops"
#include "pair"
#include "bits/bst.h"
#include "bits/bst_iter.h"

```

Include dependency graph for avl\_tree:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [cxx::Avl\\_tree\\_node](#)

*Node of an AVL tree.*

- [class cxx::Avl\\_tree< Node, Get\\_key, Compare >](#)

*A generic AVL tree.*

## Namespaces

- [CXX](#)

*Our C++ library.*

## 15.97.1 Detailed Description

AVL tree.

Definition in file [avl\\_tree](#).

## 15.98 avl\_tree

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include "std_ops"
00028 #include "pair"
00029
00030 #include "bits/bst.h"
00031 #include "bits/bst_iter.h"
00032
00033 namespace cxx {
00034
00038 class Avl_tree_node : public Bits::Bst_node
00039 {
00040 private:
00041 template< typename Node, typename Get_key, typename Compare >
00042 friend class Avl_tree;
00043
00045 typedef Bits::Direction Bal;
00047 typedef Bits::Direction Dir;
00048
00049 // We are a final BST node, hide interior.
00051 using Bits::Bst_node::next;
00052 using Bits::Bst_node::next_p;
00053 using Bits::Bst_node::rotate;
00056 Bal _balance;
00058
00059 protected:
00061 Avl_tree_node() {}
00062
00063 private:
00065 Avl_tree_node(Avl_tree_node const &o);
00066
00068 void operator = (Avl_tree_node const &o)

```

```

00069 {
00070 Bits::Bst_node::operator = (o);
00071 _balance = o._balance;
00072 }
00073
00075 explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(
Dir::N) {}
00076
00078 static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);
00079
00081 bool balanced() const { return _balance == Bal::N; }
00082
00084 Bal balance() const { return _balance; }
00085
00087 void balance(Bal b) { _balance = b; }
00088 };
00089
00090
00107 template< typename Node, typename Get_key,
00108 typename Compare = Lt_functor<typename Get_key::Key_type> >
00109 class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
00110 {
00111 private:
00112 typedef Bits::Bst<Node, Get_key, Compare> Bst;
00113
00115 using Bst::_head;
00116
00118 using Bst::k;
00119
00121 typedef typename Avl_tree_node::Bal Bal;
00123 typedef typename Avl_tree_node::Bal Dir;
00124
00126 Avl_tree(Avl_tree const &o);
00127
00129 void operator = (Avl_tree const &o);
00130
00131 public:
00133 typedef typename Bst::Key_type Key_type;
00134 typedef typename Bst::Key_param_type Key_param_type;
00136
00137 // Grab iterator types from Bst
00139 typedef typename Bst::Iterator Iterator;
00142 typedef typename Bst::Const_iterator Const_iterator;
00144 typedef typename Bst::Rev_iterator Rev_iterator;
00146 typedef typename Bst::Const_rev_iterator
Const_rev_iterator;
00148
00156 Pair<Node *, bool> insert(Node *new_node);
00157
00164 Node *remove(Key_param_type key);
00168 Node *erase(Key_param_type key) { return remove(key); }
00169
00171 Avl_tree() : Bst() {}
00173 ~Avl_tree() noexcept
00174 {
00175 this->remove_all([](Node *){});
00176 }
00177
00178 #ifdef __DEBUG_L4_AVL
00179 bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
00180 bool rec_dump(bool print)
00181 {
00182 int dp=0;
00183 return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
00184 }
00185 #endif
00186 };
00187
00188
00189 //-----
00190 /* IMPLEMENTATION: Bits::__Bst_iter_b */
00191
00192
00193 inline
00194 Bits::Bst_node *
00195 Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
00196 {
00197 typedef Bits::Bst_node N;
00198 typedef Avl_tree_node A;
00199 N *tmp[2] = { *t, N::next(*t, idir) };
00200 *t = N::next(tmp[1], !idir);
00201 A *n = static_cast<A*>(*t);
00202
00203 N::next(tmp[0], idir, N::next(n, !idir));
00204 N::next(tmp[1], !idir, N::next(n, idir));
00205 N::next(n, !idir, tmp[0]);
00206 N::next(n, idir, tmp[1]);

```

```

00207
00208 n->balance(Bal::N);
00209
00210 if (pre == Bal::N)
00211 {
00212 static_cast<A*>(tmp[0])->balance(Bal::N);
00213 static_cast<A*>(tmp[1])->balance(Bal::N);
00214 return 0;
00215 }
00216
00217 static_cast<A*>(tmp[pre != idir])->balance(!pre);
00218 static_cast<A*>(tmp[pre == idir])->balance(Bal::N);
00219
00220 return N::next(tmp[pre == idir], !pre);
00221 }
00222
00223 //-----
00224 /* Implementation of AVL Tree */
00225
00226 /* Insert new _Node. */
00227 template< typename Node, typename Get_key, class Compare>
00228 Pair<Node *, bool>
00229 Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
00230 {
00231 typedef Avl_tree_node A;
00232 typedef Bits::Bst_node N;
00233 N **t = &_head; /* search variable */
00234 N **s = &_head; /* node where rebalancing may occur */
00235 Key_param_type new_key = Get_key::key_of(new_node);
00236
00237 // search insertion point
00238 for (N *p; (p = *t);)
00239 {
00240 Dir b = this->dir(new_key, p);
00241 if (b == Dir::N)
00242 return pair(static_cast<Node*>(p), false);
00243
00244 if (!static_cast<A const *>(p)->balanced())
00245 s = t;
00246
00247 t = N::next_p(p, b);
00248 }
00249
00250 *static_cast<A*>(new_node) = A(true);
00251 *t = new_node;
00252
00253 N *n = *s;
00254 A *a = static_cast<A*>(n);
00255 if (!a->balanced())
00256 {
00257 A::Bal b(this->greater(new_key, n));
00258 if (a->balance() != b)
00259 {
00260 // ok we got in balance the shorter subtree go higher
00261 a->balance(Bal::N);
00262 // propagate the new balance down to the new node
00263 n = N::next(n, b);
00264 }
00265 else if (b == Bal(this->greater(new_key, N::next(n, b))))
00266 {
00267 // left-left or right-right case -> single rotation
00268 A::rotate(s, b);
00269 a->balance(Bal::N);
00270 static_cast<A*>(*s)->balance(Bal::N);
00271 n = N::next(*s, b);
00272 }
00273 else
00274 {
00275 // need a double rotation
00276 n = N::next(N::next(n, b), !b);
00277 n = A::rotate2(s, b, n == new_node ? Bal::N : Bal(this->greater(new_key, n)));
00278 }
00279 }
00280
00281 for (A::Bal b; n && n != new_node; static_cast<A*>(n)->balance(b), n = N::next(n, b))
00282 b = Bal(this->greater(new_key, n));
00283
00284 return pair(new_node, true);
00285 }
00286
00287 /* remove an element */
00288 template< typename Node, typename Get_key, class Compare>
00289 inline
00290 Node *Avl_tree<Node, Get_key, Compare>::remove(Key_param_type key)
00291 {
00292 typedef Avl_tree_node A;

```



```

00294 typedef Bits::Bst_node N;
00295 N **q = &_head; /* search variable */
00296 N **s = &_head; /* last ('deepest') node on the search path to q
00297 * with balance 0, at this place the rebalancing
00298 * stops in any case */
00299 N **t = 0;
00300 Dir dir;
00301
00302 // find target node and rebalancing entry
00303 for (N *n; (n = *q); q = N::next_p(n, dir))
00304 {
00305 dir = Dir(this->greater(key, n));
00306 if (dir == Dir::L && !this->greater(k(n), key))
00307 /* found node */
00308 t = q;
00309
00310 if (!N::next(n, dir))
00311 break;
00312
00313 A const *a = static_cast<A const *>(n);
00314 if (a->balanced() || (a->balance() == !dir && N::next<A>(n, !dir)->balanced()))
00315 s = q;
00316 }
00317
00318 // nothing found
00319 if (!t)
00320 return 0;
00321
00322 A *i = static_cast<A*>(*t);
00323
00324 for (N *n; (n = *s); s = N::next_p(n, dir))
00325 {
00326 dir = Dir(this->greater(key, n));
00327
00328 if (!N::next(n, dir))
00329 break;
00330
00331 A *a = static_cast<A*>(n);
00332 // got one out of balance
00333 if (a->balanced())
00334 a->balance(!dir);
00335 else if (a->balance() == dir)
00336 a->balance(Bal::N);
00337 else
00338 {
00339 // we need rotations to get in balance
00340 Bal b = N::next<A>(n, !dir)->balance();
00341 if (b == dir)
00342 A::rotate2(s, !dir, N::next<A>(N::next(n, !dir), dir)->balance());
00343 else
00344 {
00345 A::rotate(s, !dir);
00346 if (b != Bal::N)
00347 {
00348 a->balance(Bal::N);
00349 static_cast<A*>(*s)->balance(Bal::N);
00350 }
00351 else
00352 {
00353 a->balance(!dir);
00354 static_cast<A*>(*s)->balance(dir);
00355 }
00356 }
00357 if (n == i)
00358 t = N::next_p(*s, dir);
00359 }
00360 }
00361
00362 A *n = static_cast<A*>(*q);
00363 *t = n;
00364 *q = N::next(n, !dir);
00365 *n = *l;
00366
00367 return static_cast<Node*>(i);
00368 }
00369
00370 #ifdef __DEBUG_L4_AVL
00371 template< typename Node, typename Get_key, class Compare>
00372 bool Avl_tree<Node, Get_key, Compare>::rec_dump(
00373 Avl_tree_node *n, int depth, int *dp, bool print, char pfx)
00374 {
00375 typedef Avl_tree_node A;
00376
00377 if (!n)
00378 return true;
00379
00380 int dpx[2] = {depth, depth};

```



## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

## Variables

- IOModifier const [L4::hex](#)  
*Modifies the stream to print numbers as hexadecimal values.*
- IOModifier const [L4::dec](#)  
*Modifies the stream to print numbers as decimal values.*

### 15.99.1 Detailed Description

Basic IO stream.

Definition in file [basic\\_ostream](#).

## 15.100 basic\_ostream

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace L4 {
00026
00033 class IOModifier
00034 {
00035 public:
00036 IOModifier(int x) : mod(x) {}
00037 bool operator == (IOModifier o) { return mod == o.mod; }
00038 bool operator != (IOModifier o) { return mod != o.mod; }
00039 int mod;
00040 };
00041
00046 class IOBackend
00047 {
00048 public:
00049 typedef int Mode;
00050
00051 protected:
00052 friend class BasicOStream;
00053
00054 IOBackend()
00055 : int_mode(10)
00056 {}
00057
00058 virtual ~IOBackend() {}
00059
00060 virtual void write(char const *str, unsigned len) = 0;
00061
00062 private:

```

```

00063 void write(IOModifier m);
00064 void write(long long int c, int len);
00065 void write(long long unsigned c, int len);
00066 void write(long long unsigned c, unsigned char base = 10,
00067 unsigned char len = 0, char pad = ' ');
00068
00069 Mode mode() const
00070 { return int_mode; }
00071
00072 void mode(Mode m)
00073 { int_mode = m; }
00074
00075 int int_mode;
00076 };
00077
00082 class BasicOStream
00083 {
00084 public:
00085 BasicOStream(IOBackend *b)
00086 : iob(b)
00087 {}
00088
00089 void write(char const *str, unsigned len)
00090 {
00091 if (iob)
00092 iob->write(str, len);
00093 }
00094
00095 void write(long long int c, int len)
00096 {
00097 if (iob)
00098 iob->write(c, len);
00099 }
00100
00101 void write(long long unsigned c, unsigned char base = 10,
00102 unsigned char len = 0, char pad = ' ')
00103 {
00104 if (iob)
00105 iob->write(c, base, len, pad);
00106 }
00107
00108 void write(long long unsigned c, int len)
00109 {
00110 if (iob)
00111 iob->write(c, len);
00112 }
00113
00114 void write(IOModifier m)
00115 {
00116 if (iob)
00117 iob->write(m);
00118 }
00119
00120 IOBackend::Mode be_mode() const
00121 {
00122 if (iob)
00123 return iob->mode();
00124 return 0;
00125 }
00126
00127 void be_mode(IOBackend::Mode m)
00128 {
00129 if (iob)
00130 iob->mode(m);
00131 }
00132 private:
00133 IOBackend *iob;
00134 };
00135
00141 class IONumFmt
00142 {
00143 public:
00144 IONumFmt(unsigned long long n, unsigned char base = 10,
00145 unsigned char len = 0, char pad = ' ')
00146 : n(n), base(base), len(len), pad(pad)
00147 {}
00148
00149 BasicOStream &print(BasicOStream &o) const;
00150
00151 private:
00152 unsigned long long n;
00153 unsigned char base, len;
00154 char pad;
00155 };
00156
00157 inline IONumFmt n_hex(unsigned long long n) { return IONumFmt(n, 16); }

```

```

00158
00162 extern IOModifier const hex;
00163
00167 extern IOModifier const dec;
00168
00169 inline
00170 BasicOStream &IONumFmt::print(BasicOStream &o) const
00171 {
00172 o.write(n, base, len, pad);
00173 return o;
00174 }
00175 }
00176
00177 // Implementation
00178
00179 inline
00180 L4::BasicOStream &
00181 operator << (L4::BasicOStream &s, char const * const str)
00182 {
00183 if (!str)
00184 {
00185 s.write("(NULL)", 6);
00186 return s;
00187 }
00188 unsigned l = 0;
00189 for (; str[l] != 0; l++)
00190 {
00191 s.write(str, l);
00192 return s;
00193 }
00194 inline
00195 L4::BasicOStream &
00196 operator << (L4::BasicOStream &s, signed short u)
00197 {
00198 s.write((long long signed)u, -1);
00199 return s;
00200 }
00201 inline
00202 L4::BasicOStream &
00203 operator << (L4::BasicOStream &s, signed u)
00204 {
00205 s.write((long long signed)u, -1);
00206 return s;
00207 }
00208 inline
00209 L4::BasicOStream &
00210 operator << (L4::BasicOStream &s, signed long u)
00211 {
00212 s.write((long long signed)u, -1);
00213 return s;
00214 }
00215 inline
00216 L4::BasicOStream &
00217 operator << (L4::BasicOStream &s, signed long long u)
00218 {
00219 s.write(u, -1);
00220 return s;
00221 }
00222 inline
00223 L4::BasicOStream &
00224 operator << (L4::BasicOStream &s, unsigned short u)
00225 {
00226 s.write((long long unsigned)u, -1);
00227 return s;
00228 }
00229 inline
00230 L4::BasicOStream &
00231 operator << (L4::BasicOStream &s, unsigned u)
00232 {
00233 s.write((long long unsigned)u, -1);
00234 return s;
00235 }
00236 inline
00237 L4::BasicOStream &
00238 operator << (L4::BasicOStream &s, unsigned long u)
00239 {
00240 s.write((long long unsigned)u, -1);
00241 return s;
00242 }
00243 inline
00244 L4::BasicOStream &
00245 operator << (L4::BasicOStream &s, unsigned long long u)
00246 {
00247 s.write((long long unsigned)u, -1);
00248 return s;
00249 }
00250

```

```

00251 }
00252
00253 inline
00254 L4::BasicOStream &
00255 operator << (L4::BasicOStream &s, unsigned long long u)
00256 {
00257 s.write(u, -1);
00258 return s;
00259 }
00260
00261 inline
00262 L4::BasicOStream &
00263 operator << (L4::BasicOStream &s, void const *u)
00264 {
00265 long unsigned x = (long unsigned)u;
00266 L4::IOBackend::Mode mode = s.be_mode();
00267 s.write(L4::hex);
00268 s.write((long long unsigned)x, -1);
00269 s.be_mode(mode);
00270 return s;
00271 }
00272
00273 inline
00274 L4::BasicOStream &
00275 operator << (L4::BasicOStream &s, L4::IOModifier m)
00276 {
00277 s.write(m);
00278 return s;
00279 }
00280
00281 inline
00282 L4::BasicOStream &
00283 operator << (L4::BasicOStream &s, char c)
00284 {
00285 s.write(&c, 1);
00286 return s;
00287 }
00288
00289 inline
00290 L4::BasicOStream &
00291 operator << (L4::BasicOStream &o, L4::IONumFmt const &n)
00292 { return n.print(o); }

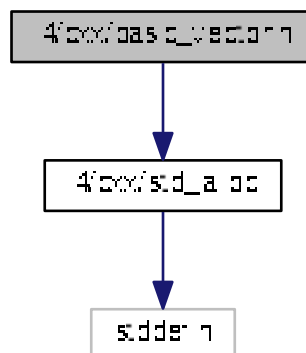
```

## 15.101 l4/cxx/basic\_vector.h File Reference

Basic vector.

```
#include <l4/cxx/std_alloc>
```

Include dependency graph for basic\_vector.h:



## Namespaces

- [cxx](#)

*Our C++ library.*

### 15.101.1 Detailed Description

Basic vector.

Definition in file [basic\\_vector.h](#).

## 15.102 basic\_vector.h

```

00001
00002 /*
00003 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004 * economic rights: Technische Universität Dresden (Germany)
00005 *
00006 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU General Public License 2.
00008 * Please see the COPYING-GPL-2 file for details.
00009 *
00010 * As a special exception, you may use this file as part of a free software
00011 * library without restriction. Specifically, if other files instantiate
00012 * templates or use macros or inline functions from this file, or you compile
00013 * this file and link it with other files to produce an executable, this
00014 * file does not by itself cause the resulting executable to be covered by
00015 * the GNU General Public License. This exception does not however
00016 * invalidate any other reasons why the executable file might be covered by
00017 * the GNU General Public License.
00018 */
00019 #pragma once
00020
00021 #include <l4/cxx/std_alloc>
00022
00023 namespace cxx {
00024
00025 template< typename T >
00026 class Basic_vector
00027 {
00028 public:
00029 Basic_vector(T *array, unsigned long capacity)
00030 : _array(array), _capacity(capacity)
00031 {
00032 for (unsigned long i = 0; i < capacity; ++i)
00033 new (&_array[i]) T();
00034 }
00035 private:
00036 T *_array;
00037 unsigned long _capacity;
00038 };
00039
00040

```

## 15.103 l4/cxx/bits/bst.h File Reference

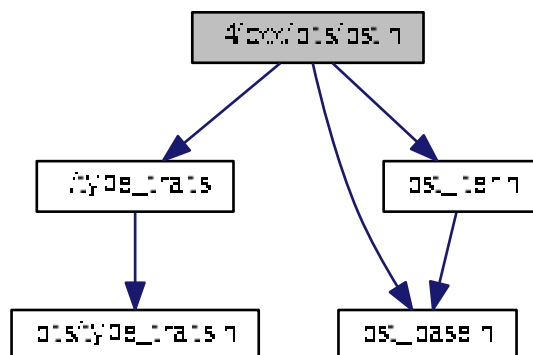
AVL tree.

```

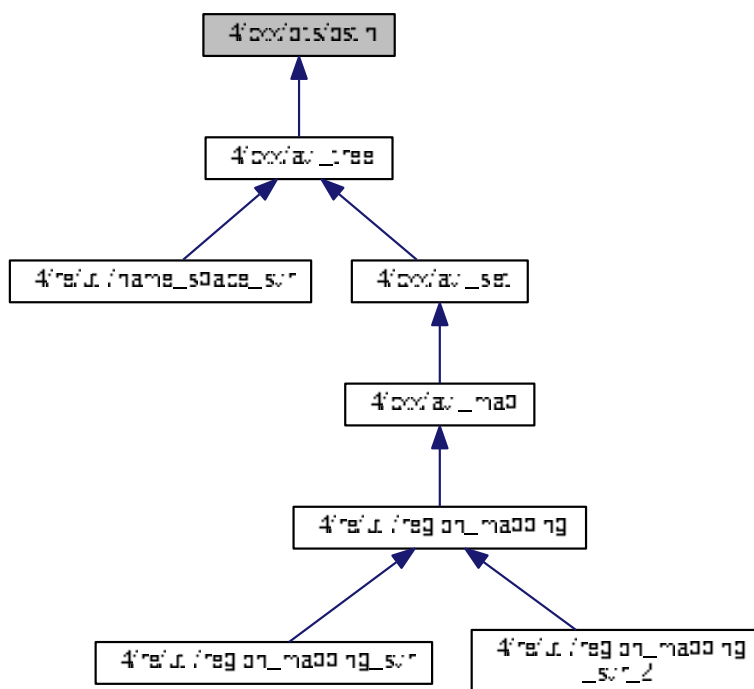
#include "../type_traits"
#include "bst_base.h"

```

```
#include "bst_iter.h"
Include dependency graph for bst.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `cxx::Bits::Bst< Node, Get_key, Compare >`  
Basic binary search tree (BST).



## Namespaces

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 15.103.1 Detailed Description

AVL tree.

Definition in file [bst.h](#).

## 15.104 bst.h

```

00001 // vi:ft=cpp
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include "../type_traits"
00027 #include "bst_base.h"
00028 #include "bst_iter.h"
00029
00030 namespace cxx { namespace Bits {
00031
00039 template< typename Node, typename Get_key, typename Compare >
00040 class Bst
00041 {
00042 private:
00043 typedef Direction Dir;
00044 struct Fwd
00045 {
00046 static Node *child(Node const *n, Direction d)
00047 { return Bst_node::next<Node>(n, d); }
00048
00049 static bool cmp(Node const *l, Node const *r)
00050 { return Compare()(Get_key::key_of(l), Get_key::key_of(r)); }
00051 };
00052
00053 struct Rev
00054 {
00055 static Node *child(Node const *n, Direction d)
00056 { return Bst_node::next<Node>(n, !d); }
00057
00058 static bool cmp(Node const *l, Node const *r)
00059 { return Compare()(Get_key::key_of(r), Get_key::key_of(l)); }
00060 };
00061
00062 public:
00066 typedef typename Get_key::Key_type Key_type;
00068 typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00069
00071 typedef Fwd Fwd_iter_ops;
00073 typedef Rev Rev_iter_ops;
00074
00076

```

```

00077 typedef __Bst_iter<Node, Node, Fwd> Iterator;
00080 typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00082 typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00084 typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00086
00087 protected:
00096
00098 Bst_node *_head;
00099
00101 Bst() : _head(0) {}
00102
00104 Node *head() const { return static_cast<Node*>(_head); }
00105
00107 static Key_type k(Bst_node const *n)
00108 { return Get_key::key_of(static_cast<Node const *>(n)); }
00109
00118 static Dir dir(Key_param_type l, Key_param_type r)
00119 {
00120 Compare cmp;
00121 Dir d(cmp(r, l));
00122 if (d == Direction::L && !cmp(l, r))
00123 return Direction::N;
00124 return d;
00125 }
00126
00135 static Dir dir(Key_param_type l, Bst_node const *r)
00136 { return dir(l, k(r)); }
00137
00139 static bool greater(Key_param_type l, Key_param_type r)
00140 { return Compare()(r, l); }
00141
00143 static bool greater(Key_param_type l, Bst_node const *r)
00144 { return greater(l, k(r)); }
00145
00147 static bool greater(Bst_node const *l, Bst_node const *r)
00148 { return greater(k(l), k(r)); }
00157 template<typename FUNC>
00158 static void remove_tree(Bst_node *head, FUNC &&callback)
00159 {
00160 if (Bst_node *n = Bst_node::next(head, Dir::L))
00161 remove_tree(n, callback);
00162
00163 if (Bst_node *n = Bst_node::next(head, Dir::R))
00164 remove_tree(n, callback);
00165
00166 callback(static_cast<Node *>(head));
00167 }
00168
00169 public:
00170
00179 Const_iterator begin() const { return Const_iterator(head()); }
00184 Const_iterator end() const { return Const_iterator(); }
00185
00190 Iterator begin() { return Iterator(head()); }
00195 Iterator end() { return Iterator(); }
00196
00201 Const_rev_iterator rbegin() const { return Const_rev_iterator(
head()); }
00206 Const_rev_iterator rend() const { return Const_rev_iterator(); }
00207
00212 Rev_iterator rbegin() { return Rev_iterator(head()); }
00217 Rev_iterator rend() { return Rev_iterator(); }
00231 Node *find_node(Key_param_type key) const;
00232
00239 Node *lower_bound_node(Key_param_type key) const;
00240
00247 Const_iterator find(Key_param_type key) const;
00248
00257 template<typename FUNC>
00258 void remove_all(FUNC &&callback)
00259 {
00260 if (!_head)
00261 return;
00262
00263 Bst_node *head = _head;
00264 _head = 0;
00265 remove_tree(head, cxx::forward<FUNC>(callback));
00266 }
00267
00268
00270 };
00271
00272 /* find an element */
00273 template< typename Node, typename Get_key, class Compare>
00274 inline
00275 Node *
00276 Bst<Node, Get_key, Compare>::find_node(

```

```

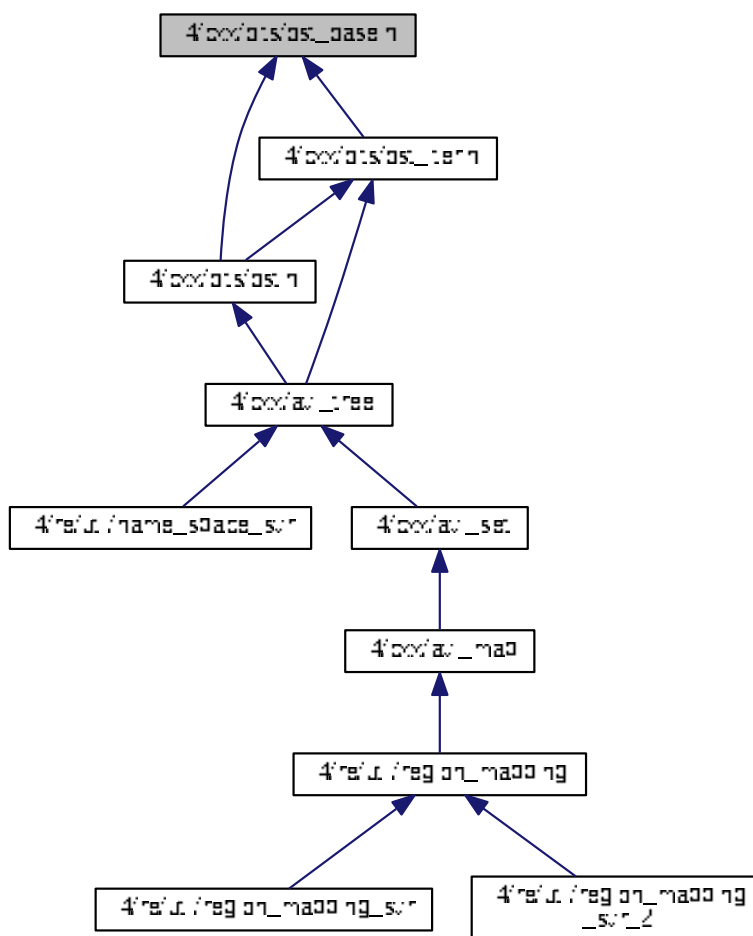
 Key_param_type key) const
00277 {
00278 Dir d;
00279
00280 for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00281 {
00282 d = dir(key, q);
00283 if (d == Dir::N)
00284 return static_cast<Node*>(q);
00285 }
00286 return 0;
00287 }
00288
00289 template< typename Node, typename Get_key, class Compare>
00290 inline
00291 Node *
00292 Bst<Node, Get_key, Compare>::lower_bound_node(
 Key_param_type key) const
00293 {
00294 Dir d;
00295 Bst_node *r = 0;
00296
00297 for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00298 {
00299 d = dir(key, q);
00300 if (d == Dir::L)
00301 r = q; // found a node greater than key
00302 else if (d == Dir::N)
00303 return static_cast<Node*>(q);
00304 }
00305 return static_cast<Node*>(r);
00306 }
00307
00308 /* find an element */
00309 template< typename Node, typename Get_key, class Compare>
00310 inline
00311 typename Bst<Node, Get_key, Compare>::Const_iterator
00312 Bst<Node, Get_key, Compare>::find(
 Key_param_type key) const
00313 {
00314 Bst_node *q = _head;
00315 Bst_node *r = q;
00316
00317 for (Dir d; q; q = Bst_node::next(q, d))
00318 {
00319 d = dir(key, q);
00320 if (d == Dir::N)
00321 return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));
00322
00323 if (d != Dir::L && q == r)
00324 r = Bst_node::next(q, d);
00325 }
00326 return Iterator();
00327 }
00328
00329 }}

```

## 15.105 l4/cxx/bits/bst\_base.h File Reference

AVL tree.

This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [cxx::Bits::Direction](#)  
*The direction to go in a binary search tree.*
- class [cxx::Bits::Bst\\_node](#)  
*Basic type of a node in a binary search tree (BST).*

## Namespaces

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 15.105.1 Detailed Description

AVL tree.

Definition in file [bst\\_base.h](#).

## 15.106 bst\_base.h

```

00001 // vi:ft=cpp
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 /*
00028 * This file contains very basic bits for implementing binary search trees
00029 */
00030 namespace cxx {
00034 namespace Bits {
00035
00039 struct Direction
00040 {
00042 enum Direction_e
00043 {
00044 L = 0,
00045 R = 1,
00046 N = 2
00047 };
00048 unsigned char d;
00049
00051 Direction() {}
00052
00054 Direction(Direction_e d) : d(d) {}
00055
00057 explicit Direction(bool b) : d(Direction_e(b)) /*d(b ? R : L)*/ {}
00058
00063 Direction operator ! () const { return Direction(!d); }
00064
00066
00067 bool operator == (Direction_e o) const { return d == o; }
00068 bool operator != (Direction_e o) const { return d != o; }
00069 bool operator == (Direction o) const { return d == o.d; }
00070 bool operator != (Direction o) const { return d != o.d; }
00072 };
00073
00077 class Bst_node
00078 {
00079 // all BSTs are friends
00080 template< typename Node, typename Get_key, typename Compare >
00081 friend class Bst;
00082
00083 protected:
00092
00094 static Bst_node *next(Bst_node const *p, Direction d)
00095 { return p->_c[d.d]; }
00096
00098 static void next(Bst_node *p, Direction d, Bst_node *n)
00099 { p->_c[d.d] = n; }
00100
00102 static Bst_node **next_p(Bst_node *p, Direction d)
00103 { return &p->_c[d.d]; }
00104
00106 template< typename Node > static

```

```

00107 Node *next(Bst_node const *p, Direction d)
00108 { return static_cast<Node *>(p->_c[d.d]); }
00109
00111 static void rotate(Bst_node **t, Direction idir);
00114 private:
00115 Bst_node *_c[2];
00116
00117 protected:
00119 Bst_node() {}
00120
00122 explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
00123 };
00124
00125 inline
00126 void
00127 Bst_node::rotate(Bst_node **t, Direction idir)
00128 {
00129 Bst_node *tmp = *t;
00130 *t = next(tmp, idir);
00131 next(tmp, idir, next(*t, !idir));
00132 next(*t, !idir, tmp);
00133 }
00134
00135 }

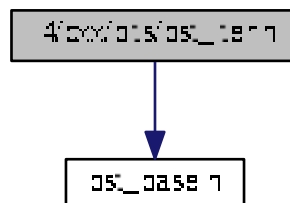
```

## 15.107 l4/cxx/bits/bst\_iter.h File Reference

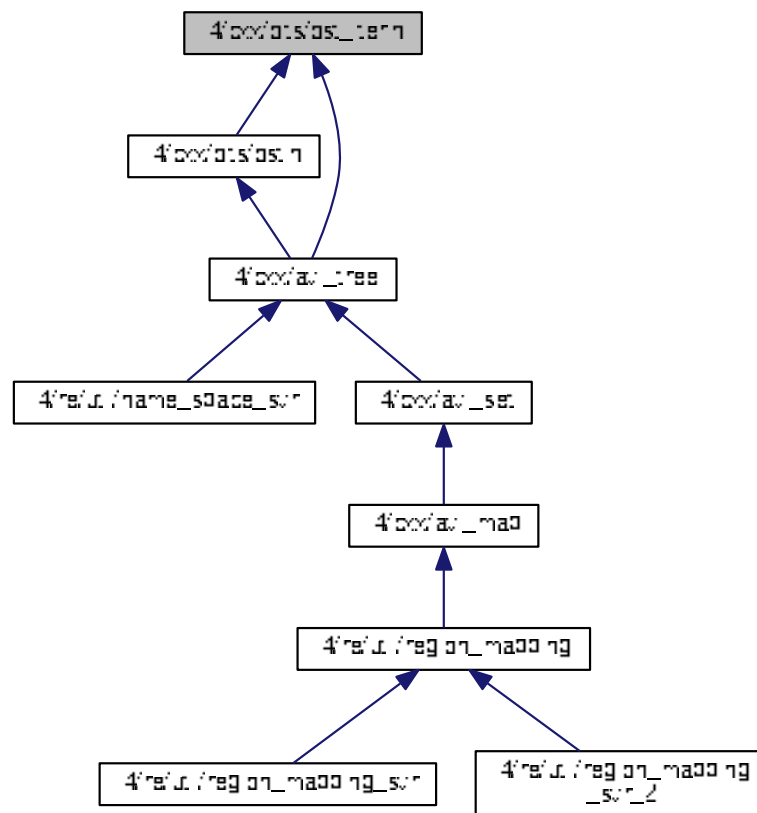
AVL tree.

```
#include "bst_base.h"
```

Include dependency graph for bst\_iter.h:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*

## 15.107.1 Detailed Description

AVL tree.

Definition in file [bst\\_iter.h](#).

## 15.108 bst\_iter.h

```

00001 // vi:ft=cpp
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include "bst_base.h"
00028
00029 namespace cxx { namespace Bits {
00030
00031 template< typename Node, typename Node_op >
00032 class __Bst_iter_b
00033 {
00034 protected:
00035 typedef Direction Dir;
00036 Node const *_n;
00037 Node const *_r;
00038
00039 __Bst_iter_b() : _n(0), _r(0) {}
00040
00041 __Bst_iter_b(Node const *t)
00042 : _n(t), _r(_n)
00043 { _downmost(); }
00044
00045 __Bst_iter_b(Node const *t, Node const *r)
00046 : _n(t), _r(r)
00047 {}
00048
00049 inline void _downmost();
00050
00051 inline void inc();
00052
00053 public:
00054 bool operator == (__Bst_iter_b const &o) const { return _n == o._n; }
00055 bool operator != (__Bst_iter_b const &o) const { return _n != o._n; }
00056 };
00057
00058 template< typename Node, typename Node_type, typename Node_op >
00059 class __Bst_iter : public __Bst_iter_b<Node, Node_op>
00060 {
00061 private:
00062 typedef __Bst_iter_b<Node, Node_op> Base;
00063
00064 using Base::_n;
00065 using Base::_r;
00066 using Base::inc;
00067
00068 public:
00069 __Bst_iter() {}
00070
00071 __Bst_iter(Node const *t) : Base(t) {}
00072 __Bst_iter(Node const *t, Node const *r) : Base(t, r) {}
00073
00074 // template<typename Key2>
00075 __Bst_iter(Base const &o) : Base(o) {}
00076
00077 Node_type &operator * () const { return *const_cast<Node *>(_n); }
00078 Node_type *operator -> () const { return const_cast<Node *>(_n); }
00079 __Bst_iter &operator ++ () { inc(); return *this; }
00080 __Bst_iter &operator ++ (int)
00081 { __Bst_iter tmp = *this; inc(); return tmp; }
00082 };
00083
00084 //-----
00085 /* IMPLEMENTATION: __Bst_iter_b */
00086
00087 template< typename Node, typename Node_op>

```



```

00136 inline
00137 void __Bst_iter_b<Node, Node_op>::_downmost()
00138 {
00139 while (_n)
00140 {
00141 Node *n = Node_op::child(_n, Dir::L);
00142 if (n)
00143 _n = n;
00144 else
00145 return;
00146 }
00147 }
00148
00149 template< typename Node, typename Node_op>
00150 void __Bst_iter_b<Node, Node_op>::inc()
00151 {
00152 if (!_n)
00153 return;
00154
00155 if (_n == _r)
00156 {
00157 _r = _n = Node_op::child(_r, Dir::R);
00158 _downmost();
00159 return;
00160 }
00161
00162 if (Node_op::child(_n, Dir::R))
00163 {
00164 _n = Node_op::child(_n, Dir::R);
00165 _downmost();
00166 return;
00167 }
00168
00169 Node const *q = _r;
00170 Node const *p = _r;
00171 while (1)
00172 {
00173 if (Node_op::cmp(_n, q))
00174 {
00175 p = q;
00176 q = Node_op::child(q, Dir::L);
00177 }
00178 else if (_n == q || Node_op::child(q, Dir::R) == _n)
00179 {
00180 _n = p;
00181 return;
00182 }
00183 else
00184 q = Node_op::child(q, Dir::R);
00185 }
00186 }
00187
00188 }}

```

## 15.109 l4/cxx/exceptions File Reference

Base exceptions.

```

#include <l4/cxx/l4types.h>
#include <l4/cxx/basic_ostream>
#include <l4/sys/err.h>
#include <l4/sys/capability>
#include <l4/util/backtrace.h>

```



- class [L4::Element\\_already\\_exists](#)  
*Exception for duplicate element insertions.*
- class [L4::Unknown\\_error](#)  
*Exception for an unknown condition.*
- class [L4::Element\\_not\\_found](#)  
*Exception for a failed lookup (element not found).*
- class [L4::Invalid\\_capability](#)  
*Indicates that an invalid object was invoked.*
- class [L4::Com\\_error](#)  
*Error conditions during IPC.*
- class [L4::Bounds\\_error](#)  
*Access out of bounds.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

## Macros

- `#define L4\_CXX\_EXCEPTION\_BACKTRACE 20`  
*Number of instruction pointers in backtrace.*

### 15.109.1 Detailed Description

Base exceptions.

Definition in file [exceptions](#).

## 15.110 exceptions

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/cxx/l4types.h>
00028 #include <l4/cxx/basic_ostream>
00029 #include <l4/sys/err.h>
00030 #include <l4/sys/capability>
00031
00032

```

```

00038
00039 #ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
00040 # define L4_CXX_EXCEPTION_BACKTRACE 20
00041 #endif
00042
00043 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00044 #include <l4/util/backtrace.h>
00045 #endif
00046
00048 namespace L4
00049 {
00062 class Exception_tracer
00063 {
00064 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00065 private:
00066 void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00067 int _frame_cnt;
00068
00069 protected:
00073 #if defined(__PIC__)
00074 Exception_tracer() throw() : _frame_cnt(0) {}
00075 #else
00076 Exception_tracer() throw()
00077 : _frame_cnt(l4util_backtrace(_pc_array,
00078 L4_CXX_EXCEPTION_BACKTRACE)) {}
00078 #endif
00079
00080 public:
00084 void const *const *_pc_array() const throw() { return _pc_array; }
00088 int frame_count() const throw() { return _frame_cnt; }
00089 #else
00090 protected:
00094 Exception_tracer() throw() {}
00095
00096 public:
00100 void const *const *_pc_array() const throw() { return 0; }
00104 int frame_count() const throw() { return 0; }
00105 #endif
00106 };
00107
00116 class Base_exception : public Exception_tracer
00117 {
00118 protected:
00120 Base_exception() throw() {}
00121
00122 public:
00126 virtual char const *str() const throw () = 0;
00127
00129 virtual ~Base_exception() throw () {}
00130 };
00131
00139 class Runtime_error : public Base_exception
00140 {
00141 private:
00142 long _errno;
00143 char _extra[80];
00144
00145 public:
00152 explicit Runtime_error(long err_no, char const *extra = 0) throw ()
00153 : _errno(err_no)
00154 {
00155 if (!extra)
00156 _extra[0] = 0;
00157 else
00158 {
00159 unsigned i = 0;
00160 for (; i < sizeof(_extra) && extra[i]; ++i)
00161 _extra[i] = extra[i];
00162 _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00163 }
00164 }
00165 char const *str() const throw ()
00166 { return l4sys_errtostr(_errno); }
00167
00173 char const *extra_str() const { return _extra; }
00174 ~Runtime_error() throw () {}
00175
00181 long err_no() const throw() { return _errno; }
00182 };
00183
00188 class Out_of_memory : public Runtime_error
00189 {
00190 public:
00192 explicit Out_of_memory(char const *extra = "") throw()
00193 : Runtime_error(-L4_ENOMEM, extra) {}
00195 ~Out_of_memory() throw() {}
00196 };

```

```

00197
00198
00203 class Element_already_exists : public Runtime_error
00204 {
00205 public:
00206 explicit Element_already_exists(char const *e = "") throw()
00207 : Runtime_error(-L4_EEXIST, e) {}
00208 ~Element_already_exists() throw() {}
00209 };
00210
00219 class Unknown_error : public Base_exception
00220 {
00221 public:
00222 Unknown_error() throw() {}
00223 char const *str() const throw() { return "unknown error"; }
00224 ~Unknown_error() throw() {}
00225 };
00226
00231 class Element_not_found : public Runtime_error
00232 {
00233 public:
00234 explicit Element_not_found(char const *e = "") throw()
00235 : Runtime_error(-L4_ENOENT, e) {}
00236 };
00237
00245 class Invalid_capability : public Base_exception
00246 {
00247 private:
00248 Cap<void> const _o;
00249
00250 public:
00255 explicit Invalid_capability(Cap<void> const &o) throw() : _o(o) {}
00256 template< typename T>
00257 explicit Invalid_capability(Cap<T> const &o) throw() : _o(o.
cap()) {}
00258 char const *str() const throw() { return "invalid object"; }
00259
00264 Cap<void> const &cap() const throw() { return _o; }
00265 ~Invalid_capability() throw() {}
00266 };
00267
00274 class Com_error : public Runtime_error
00275 {
00276 public:
00281 explicit Com_error(long err) throw() : Runtime_error(err) {}
00282
00283 ~Com_error() throw() {}
00284 };
00285
00289 class Bounds_error : public Runtime_error
00290 {
00291 public:
00292 explicit Bounds_error(char const *e = "") throw()
00293 : Runtime_error(-L4_ERANGE, e) {}
00294 ~Bounds_error() throw() {}
00295 };
00297 };
00298
00299 inline
00300 L4::BasicOStream &
00301 operator << (L4::BasicOStream &o, L4::Base_exception const &e)
00302 {
00303 o << "Exception: " << e.str() << ", backtrace ...\n";
00304 for (int i = 0; i < e.frame_count(); ++i)
00305 o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00306
00307 return o;
00308 }
00309
00310 inline
00311 L4::BasicOStream &
00312 operator << (L4::BasicOStream &o, L4::Runtime_error const &e)
00313 {
00314 o << "Exception: " << e.str() << ": ";
00315 if (e.extra_str())
00316 o << e.extra_str() << ": ";
00317 o << "backtrace ...\n";
00318 for (int i = 0; i < e.frame_count(); ++i)
00319 o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\n';
00320
00321 return o;
00322 }

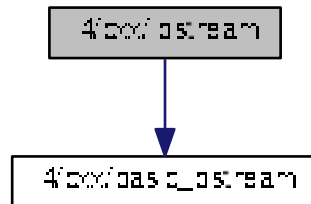
```

## 15.111 l4/cxx/iostream File Reference

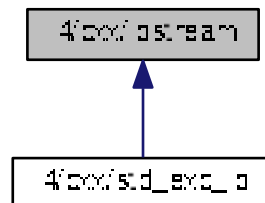
IO Stream.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for iostream:



This graph shows which files directly or indirectly include this file:



### Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### Variables

- BasicOStream [L4::cout](#)  
*Standard output stream.*
- BasicOStream [L4::cerr](#)  
*Standard error stream.*

### 15.111.1 Detailed Description

IO Stream.

Definition in file [iostream](#).

## 15.112 iostream

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/cxx/basic_ostream>
00028
00029 namespace L4 {
00030
00034 extern BasicOStream cout;
00035
00039 extern BasicOStream cerr;
00040
00041 extern void iostream_init();
00042
00043 static void __attribute__((used, constructor)) __iostream_init()
00044 { iostream_init(); }
00045 };

```

### 15.113 l4/cxx/ipc\_helper File Reference

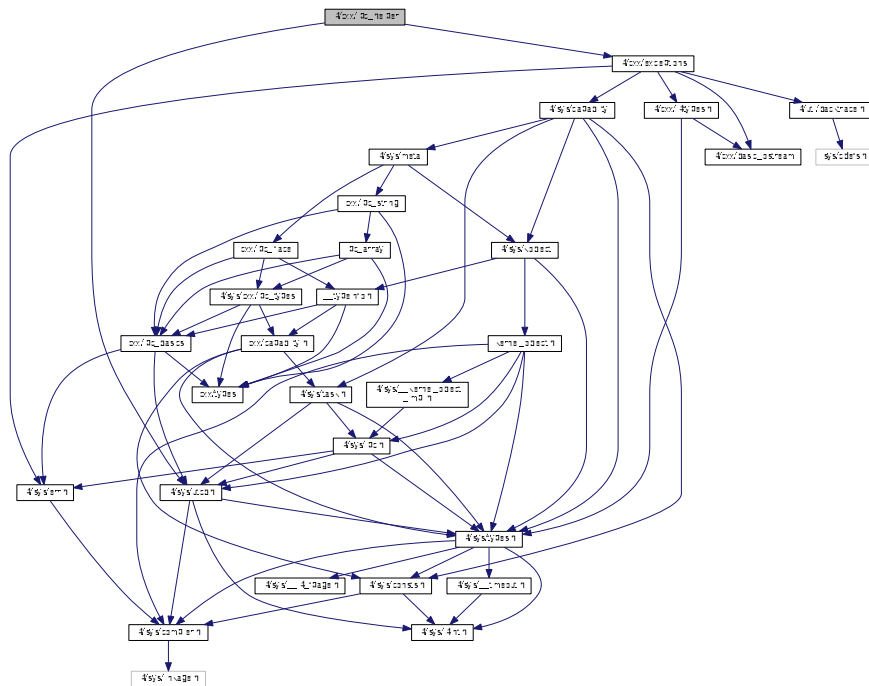
IPC helper.

```

#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>

```

Include dependency graph for `ipc_helper`:



## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

## Functions

- `void L4::throw\_ipc\_exception (L4::Cap< void > const &o, l4\_msgtag\_t const &err, l4\_utcb\_t *utcb)`  
Throw an [L4](#) IPC error as exception.
- `void L4::throw\_ipc\_exception (void const *o, l4\_msgtag\_t const &err, l4\_utcb\_t *utcb)`  
Throw an [L4](#) IPC error as exception.

### 15.113.1 Detailed Description

IPC helper.

Definition in file [ipc\\_helper](#).



## 15.114 ipc\_helper

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/cxx/exceptions>
00027 #include <l4/sys/utcb.h>
00028
00029 namespace L4
00030 {
00031 #ifdef __EXCEPTIONS
00032
00040 inline void
00041 throw_ipc_exception(L4::Cap<void> const &o,
00042 l4_msgtag_t const &err,
00043 l4_utcb_t *utcb)
00044 {
00045 (void)o;
00046 if (err.has_error())
00047 throw (L4::Com_error(l4_error_u(err, utcb)));
00048 }
00049
00057 inline void
00058 throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00059 l4_utcb_t *utcb)
00060 { throw_ipc_exception(L4::Cap<void>(o), err, utcb); }
00061 #endif
00062
00063 }

```

## 15.115 l4/cxx/ipc\_stream File Reference

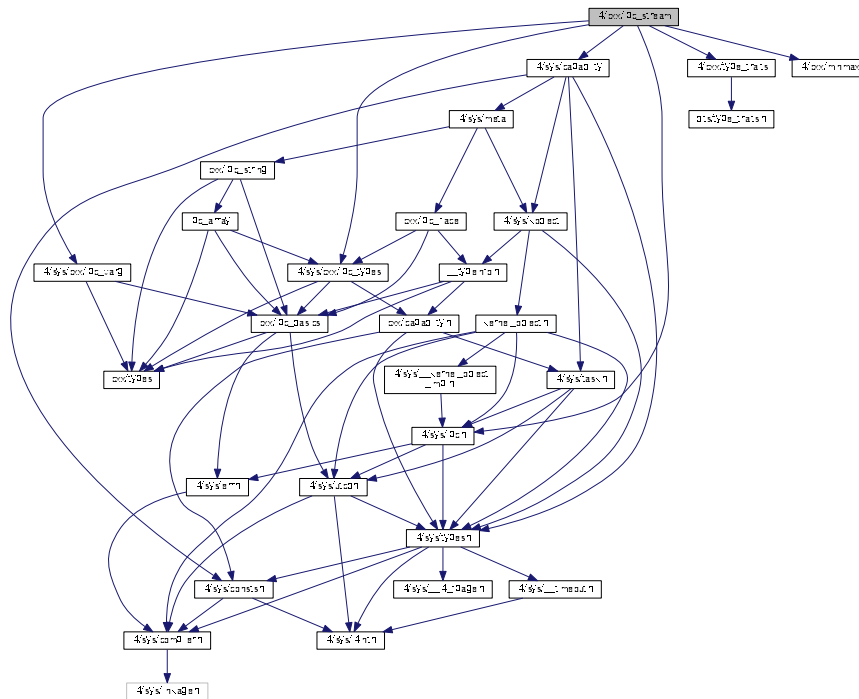
IPC stream.

```

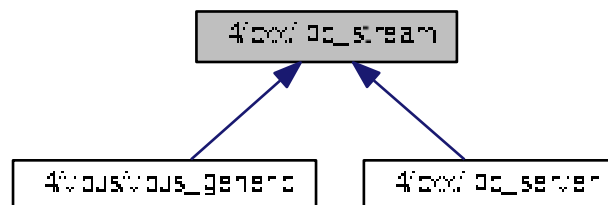
#include <l4/sys/ipc.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_varg>
#include <l4/cxx/type_traits>
#include <l4/cxx/minmax>

```

Include dependency graph for `ipc_stream`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::ipc::Str_cp_in< T >`  
Abstraction for extracting a zero-terminated string from an `ipc::lstream`.
- class `L4::ipc::Msg_ptr< T >`  
Pointer to an element of type `T` in an `ipc::lstream`.
- class `L4::ipc::lstream`  
Input stream for IPC unmarshalling.
- class `L4::ipc::ostream`  
Output stream for IPC marshalling.
- class `L4::ipc::iostream`  
Input/Output stream for IPC [un]marshalling.

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*
- [L4::lpc](#)  
*IPC related functionality.*

## Functions

- `template<typename T >`  
`Internal::Buf_cp_out< T > L4::lpc::buf_cp_out (T const *v, unsigned long size)`  
*Insert an array into an [lpc::Ostream](#).*
- `template<typename T >`  
`Internal::Buf_cp_in< T > L4::lpc::buf_cp_in (T *v, unsigned long &size)`  
*Extract an array from an [lpc::Istream](#).*
- `template<typename T >`  
`Str_cp_in< T > L4::lpc::str_cp_in (T *v, unsigned long &size)`  
*Create a [Str\\_cp\\_in](#) for the given values.*
- `template<typename T >`  
`Msg_ptr< T > L4::lpc::msg_ptr (T *&p)`  
*Create an [Msg\\_ptr](#) to adjust the given pointer.*
- `template<typename T >`  
`Internal::Buf_in< T > L4::lpc::buf_in (T *&v, unsigned long &size)`  
*Return a pointer to stream array data.*
- `L4::lpc::Istream & operator>> (L4::lpc::Istream &s, bool &v)`  
*Extract one element of type *T* from the stream *s*.*
- `L4::lpc::Istream & operator>> (L4::lpc::Istream &s, l4_msgtag_t &v)`  
*Extract the *L4* message tag from the stream *s*.*
- `template<typename T >`  
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Internal::Buf_in< T > const &v)`  
*Extract an array of *T* elements from the stream *s*.*
- `template<typename T >`  
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Msg_ptr< T > const &v)`  
*Extract an element of type *T* from the stream *s*.*
- `template<typename T >`  
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Internal::Buf_cp_in< T > const &v)`  
*Extract an array of *T* elements from the stream *s*.*
- `template<typename T >`  
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Str_cp_in< T > const &v)`  
*Extract a zero-terminated string from the stream.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, bool v)`  
*Insert an element to type *T* into the stream *s*.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, l4_msgtag_t const &v)`  
*Insert the *L4* message tag into the stream *s*.*
- `template<typename T >`  
`L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, L4::lpc::Internal::Buf_cp_out< T > const &v)`  
*Insert an array with elements of type *T* into the stream *s*.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, char const *v)`  
*Insert a zero terminated character string into the stream *s*.*
- `template<typename T >`  
`T L4::lpc::read (Istream &s)`  
*Read a value out of a stream.*

### 15.115.1 Detailed Description

IPC stream.

Definition in file [ipc\\_stream](#).

### 15.115.2 Function Documentation

#### 15.115.2.1 `operator<<()` [1/4]

```
L4::Ipc::Ostream& operator<< (
 L4::Ipc::Ostream & s,
 bool v) [inline]
```

Insert an element to type T into the stream s.

#### Parameters

|   |                                     |
|---|-------------------------------------|
| s | The stream to insert the element v. |
| v | The element to insert.              |

#### Returns

The stream s.

Definition at line 1195 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Referenced by [operator>>\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.115.2.2 operator<<() [2/4]

```

L4::Ipc::Ostream& operator<< (
 L4::Ipc::Ostream & s,
 l4_msgtag_t const & v) [inline]

```

Insert the [L4](#) message tag into the stream *s*.

#### Parameters

|          |                                               |
|----------|-----------------------------------------------|
| <i>s</i> | The stream to insert the tag <i>v</i> .       |
| <i>v</i> | The <a href="#">L4</a> message tag to insert. |

#### Returns

The stream *s*.

#### Note

Only one message tag can be inserted into a stream. Multiple insertions simply overwrite previous insertions.

Definition at line [1230](#) of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::tag\(\)](#).

Here is the call graph for this function:



**15.115.2.3** `operator<<()` [3/4]

```
template<typename T >
L4::Ipc::Ostream& operator<< (
 L4::Ipc::Ostream & s,
 L4::Ipc::Internal::Buf_cp_out< T > const & v) [inline]
```

Insert an array with elements of type T into the stream s.

**Parameters**

|   |                                                            |
|---|------------------------------------------------------------|
| s | The stream to insert the array v.                          |
| v | The array to insert (see <code>lpc::Buf_cp_out()</code> ). |

**Returns**

the stream s.

Definition at line 1246 of file `ipc_stream`.

References `L4::lpc::Ostream::put()`.

Here is the call graph for this function:

**15.115.2.4** `operator<<()` [4/4]

```
L4::Ipc::Ostream& operator<< (
 L4::Ipc::Ostream & s,
 char const * v) [inline]
```

Insert a zero terminated character string into the stream s.

**Parameters**

|   |                                    |
|---|------------------------------------|
| s | The stream to insert the string v. |
| v | The string to insert.              |

**Returns**

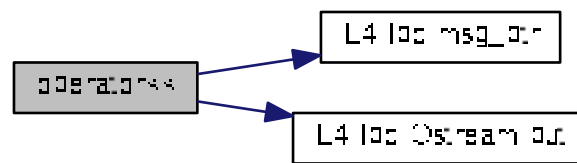
The stream `s`.

This operator produces basically the same content as the array insertion, however the length of the array is calculated using `strlen(v) + 1`. The string is copied into the message including the trailing zero.

Definition at line 1267 of file `ipc_stream`.

References [L4\\_DEPRECATED](#), [L4::Ipc::msg\\_ptr\(\)](#), and [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:

**15.115.2.5 operator>>()** [1/6]

```

L4::Ipc::Istream& operator>> (
 L4::Ipc::Istream & s,
 bool & v) [inline]

```

Extract one element of type `T` from the stream `s`.

**Parameters**

|                  |                |                             |
|------------------|----------------|-----------------------------|
|                  | <code>s</code> | The stream to extract from. |
| <code>out</code> | <code>v</code> | Extracted value.            |

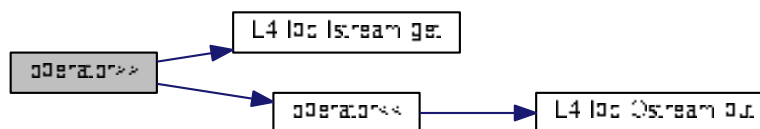
**Returns**

The stream `s`.

Definition at line 1042 of file `ipc_stream`.

References [L4::Ipc::Istream::get\(\)](#), and [operator<<\(\)](#).

Here is the call graph for this function:



#### 15.115.2.6 operator>>() [2/6]

```

L4::Ipc::Istream& operator>> (
 L4::Ipc::Istream & s,
 l4_msgtag_t & v) [inline]

```

Extract the [L4](#) message tag from the stream *s*.

##### Parameters

|     |          |                             |
|-----|----------|-----------------------------|
|     | <i>s</i> | The stream to extract from. |
| out | <i>v</i> | The extracted tag.          |

##### Returns

The stream *s*.

Definition at line [1076](#) of file [ipc\\_stream](#).

References [L4::ipc::Istream::tag\(\)](#).

Here is the call graph for this function:





15.115.2.7 `operator>>()` [3/6]

```
template<typename T >
L4::Ipc::Istream& operator>> (
 L4::Ipc::Istream & s,
 L4::Ipc::Internal::Buf_in< T > const & v) [inline]
```

Extract an array of T elements from the stream s.

## Parameters

|     |   |                                                               |
|-----|---|---------------------------------------------------------------|
|     | s | The stream to extract from.                                   |
| out | v | Pointer to the extracted array ( <code>ipc_buf_in()</code> ). |

## Returns

The stream s.

This operator actually does not copy out the data in the array, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

## Note

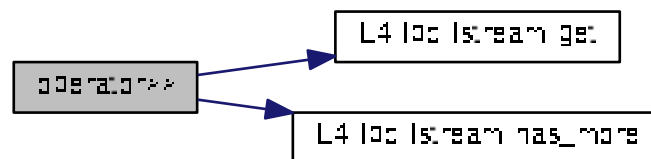
If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See `Ipc::Buf_in`, `Ipc::Buf_cp_in`, and `Ipc::Buf_cp_out`.

Definition at line 1101 of file `ipc_stream`.

References `L4::Ipc::Istream::get()`, and `L4::Ipc::Istream::has_more()`.

Here is the call graph for this function:

15.115.2.8 `operator>>()` [4/6]

```
template<typename T >
L4::Ipc::Istream& operator>> (
 L4::Ipc::Istream & s,
 L4::Ipc::Msg_ptr< T > const & v) [inline]
```

Extract an element of type T from the stream s.

**Parameters**

|     |          |                                   |
|-----|----------|-----------------------------------|
|     | <i>s</i> | The stream to extract from.       |
| out | <i>v</i> | Pointer to the extracted element. |

**Returns**

the stream *s*.

This operator actually does not copy out the data, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

See `Msg_ptr`.

Definition at line 1128 of file `ipc_stream`.

References `L4::Ipc::Istream::get()`.

Here is the call graph for this function:

**15.115.2.9 operator>>() [5/6]**

```

template<typename T >
L4::Ipc::Istream& operator>> (
 L4::Ipc::Istream & s,
 L4::Ipc::Internal::Buf_cp_in< T > const & v) [inline]

```

Extract an array of *T* elements from the stream *s*.

**Parameters**

|     |          |                                                                             |
|-----|----------|-----------------------------------------------------------------------------|
|     | <i>s</i> | The stream to extract from.                                                 |
| out | <i>v</i> | Buffer description to copy the array to ( <code>Ipc::Buf_cp_out()</code> ). |

**Returns**

The stream *s*.

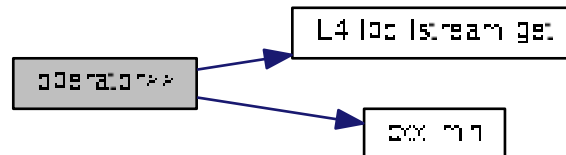
This operator does a copy out of the data into the given buffer.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1149 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `cxx::min()`.

Here is the call graph for this function:



#### 15.115.2.10 `operator>>()` [6/6]

```

template<typename T >
L4::lpc::Istream& operator>> (
 L4::lpc::Istream & s,
 L4::lpc::Str_cp_in< T > const & v) [inline]

```

Extract a zero-terminated string from the stream.

##### Parameters

|     |          |                                                                             |
|-----|----------|-----------------------------------------------------------------------------|
|     | <i>s</i> | The stream to extract from.                                                 |
| out | <i>v</i> | Buffer description to copy the array to ( <code>lpc::Str_cp_out()</code> ). |

##### Returns

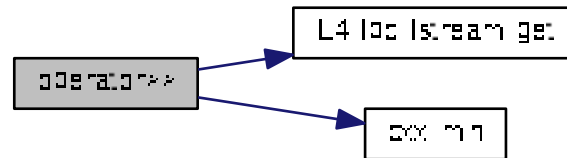
the stream *s*.

This operator does a copy out of the data into the given buffer.

Definition at line 1170 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `cxx::min()`.

Here is the call graph for this function:



## 15.116 ipc\_stream

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/ipc.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_types>
00030 #include <l4/sys/cxx/ipc_varg>
00031 #include <l4/cxx/type_traits>
00032 #include <l4/cxx/minmax>
00033
00034 #define L4_CXX_IPC_BACKWARD_COMPAT
00035
00036 namespace L4 {
00037 namespace Ipc {
00038
00039 class Ostream;
00040 class Istream;
00041
00042 namespace Internal {
00043 template< typename T >
00044 class Buf_cp_out
00045 {
00046 public:
00047 Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}
00048
00049 unsigned long size() const { return _s; }
00050
00051 T const *buf() const { return _v; }
00052
00053 private:
00054 friend class Ostream;
00055 T const *_v;
00056 unsigned long _s;
00057 };
00058
00059 template< typename T >
00060 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)

```

```

00114 { return Internal::Buf_cp_out<T>(v, size); }
00115
00116
00117 namespace Internal {
00130 template< typename T >
00131 class Buf_cp_in
00132 {
00133 public:
00142 Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00143
00144 unsigned long &size() const { return *_s; }
00145 T *buf() const { return _v; }
00146
00147 private:
00148 friend class Istream;
00149 T *_v;
00150 unsigned long *_s;
00151 };
00152 }
00153
00171 template< typename T >
00172 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00173 { return Internal::Buf_cp_in<T>(v, size); }
00174
00190 template< typename T >
00191 class Str_cp_in
00192 {
00193 public:
00202 Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00203
00204 unsigned long &size() const { return *_s; }
00205 T *buf() const { return _v; }
00206
00207 private:
00208 friend class Istream;
00209 T *_v;
00210 unsigned long *_s;
00211 };
00212
00225 template< typename T >
00226 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00227 { return Str_cp_in<T>(v, size); }
00228
00241 template< typename T >
00242 class Msg_ptr
00243 {
00244 private:
00245 T **_p;
00246 public:
00253 explicit Msg_ptr(T *&p) : _p(&p) {}
00254 void set(T *p) const { *_p = p; }
00255 };
00256
00264 template< typename T >
00265 Msg_ptr<T> msg_ptr(T *&p)
00266 { return Msg_ptr<T>(p); }
00267
00268
00269 namespace Internal {
00283 template< typename T >
00284 class Buf_in
00285 {
00286 public:
00293 Buf_in(T *&v, unsigned long &size) : _v(&v), _s(&size) {}
00294
00295 void set_size(unsigned long s) const { *_s = s; }
00296 T *&buf() const { return *_v; }
00297
00298 private:
00299 friend class Istream;
00300 T **_v;
00301 unsigned long *_s;
00302 };
00303 }
00304
00322 template< typename T >
00323 Internal::Buf_in<T> buf_in(T *&v, unsigned long &size)
00324 { return Internal::Buf_in<T>(v, size); }
00325
00326 namespace Utcb_stream_check
00327 {
00328 static bool check_utcb_data_offset(unsigned sz)
00329 { return sz > sizeof(l4_umword_t) * L4_UTCB_GENERIC_DATA_SIZE; }
00330 }
00331
00332
00347 class Istream

```

```

00348 {
00349 public:
00361 Istream(l4_utcb_t *utcb)
00362 : _tag(), _utcb(utcb),
00363 _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
00364 _pos(0), _current_buf(0)
00365 {}
00366
00371 void reset()
00372 {
00373 _pos = 0;
00374 _current_buf = 0;
00375 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00376 }
00377
00381 template< typename T >
00382 bool has_more(unsigned long count = 1)
00383 {
00384 auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(
00385 l4_umword_t);
00386 unsigned apos = cxx::Type_traits<T>::align(_pos);
00387 return (count <= max_bytes / sizeof(T))
00388 && (apos + (sizeof(T) * count)
00389 <= _tag.words() * sizeof(l4_umword_t));
00390 }
00391
00395 template< typename T >
00405 unsigned long get(T *buf, unsigned long elems)
00406 {
00407 if (L4_UNLIKELY(!has_more<T>(elems)))
00408 return 0;
00409
00410 unsigned long size = elems * sizeof(T);
00411 _pos = cxx::Type_traits<T>::align(_pos);
00412
00413 __builtin_memcpy(buf, _current_msg + _pos, size);
00414 _pos += size;
00415 return elems;
00416 }
00417
00418 template< typename T >
00424 void skip(unsigned long elems)
00425 {
00426 if (L4_UNLIKELY(!has_more<T>(elems)))
00427 return;
00428
00429 unsigned long size = elems * sizeof(T);
00430 _pos = cxx::Type_traits<T>::align(_pos);
00431 _pos += size;
00432 }
00433
00446 template< typename T >
00447 unsigned long get(Msg_ptr<T> const &buf, unsigned long elems = 1)
00448 {
00449 if (L4_UNLIKELY(!has_more<T>(elems)))
00450 return 0;
00451
00452 unsigned long size = elems * sizeof(T);
00453 _pos = cxx::Type_traits<T>::align(_pos);
00454
00455 buf.set(reinterpret_cast<T*>(_current_msg + _pos));
00456 _pos += size;
00457 return elems;
00458 }
00459
00460 template< typename T >
00469 bool get(T &v)
00470 {
00471 if (L4_UNLIKELY(!has_more<T>()))
00472 {
00473 v = T();
00474 return false;
00475 }
00476
00477 _pos = cxx::Type_traits<T>::align(_pos);
00478 v = *(reinterpret_cast<T*>(_current_msg + _pos));
00479 _pos += sizeof(T);
00480 return true;
00481 }
00482
00483 bool get(Ipc::Varg *va)
00484 {
00485 Ipc::Varg::Tag t;

```

```

00487 if (!has_more<Ipcc::Varg::Tag>())
00488 {
00489 va->tag(0);
00490 return 0;
00491 }
00492 get(t);
00493 va->tag(t);
00494 char const *d;
00495 get(msg_ptr(d), va->length());
00496 va->data(d);
00497
00498 return 1;
00499 }
00500
00510 l4_msgtag_t tag() const { return _tag; }
00511
00512 l4_msgtag_t &tag() { return _tag; }
00523
00525 inline bool put(Buf_item const &);
00530
00531 inline bool put(Small_buf const &);
00537
00538
00543
00553 inline l4_msgtag_t wait(l4_umword_t *src)
00554 { return wait(src, L4_IPC_NEVER); }
00555
00566 inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);
00567
00577 inline l4_msgtag_t receive(l4_cap_idx_t src)
00578 { return receive(src, L4_IPC_NEVER); }
00579 inline l4_msgtag_t receive(l4_cap_idx_t src,
00580 l4_timeout_t timeout);
00580
00582
00586 inline l4_utcb_t *utcb() const { return _utcb; }
00587
00588 protected:
00589 l4_msgtag_t _tag;
00590 l4_utcb_t *_utcb;
00591 char *_current_msg;
00592 unsigned _pos;
00593 unsigned char _current_buf;
00594 };
00595
00596 class Istream_copy : public Istream
00597 {
00598 private:
00599 l4_msg_regs_t _mrs;
00600
00601 public:
00602 Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.
00603 utcb()))
00604 {
00605 // do some reverse mr to utcb trickery
00606 _utcb = (l4_utcb_t *)((l4_addr_t)&mrs - (l4_addr_t)l4_utcb_mr_u((
00607 l4_utcb_t *)0));
00608 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00609 }
00610 };
00610
00625 class Ostream
00626 {
00627 public:
00631 Ostream(l4_utcb_t *utcb)
00632 : _tag(), _utcb(utcb),
00633 _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00634 _pos(0), _current_item(0)
00635 {}
00636
00640 void reset()
00641 {
00642 _pos = 0;
00643 _current_item = 0;
00644 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00645 }
00646
00654
00661 template< typename T >
00662 bool put(T *buf, unsigned long size)
00663 {
00664 size *= sizeof(T);
00665 _pos = cxx::Type_traits<T>::align(_pos);
00666 if (Utcb_stream_check::check_utcb_data_offset(_pos + size))

```

```

00667 return false;
00668
00669 __builtin_memcpy(_current_msg + _pos, buf, size);
00670 _pos += size;
00671 return true;
00672 }
00673
00674 template< typename T >
00675 bool put(T const &v)
00676 {
00677 _pos = cxx::Type_traits<T>::align(_pos);
00678 if (Utc_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00679 return false;
00680
00681 *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00682 _pos += sizeof(T);
00683 return true;
00684 }
00685
00686 int put(Varg const &va)
00687 {
00688 put(va.tag());
00689 put(va.data(), va.length());
00690
00691 return 0;
00692 }
00693
00694 template< typename T >
00695 int put(Varg_t<T> const &va)
00696 { return put(static_cast<Varg const &>(va)); }
00697
00698 l4_msgtag_t tag() const { return _tag; }
00699
00700 l4_msgtag_t &tag() { return _tag; }
00701
00702 inline bool put_snd_item(Snd_item const &);
00703
00704 inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00705
00706 inline l4_utcb_t *utcb() const { return _utcb; }
00707 #if 0
00708 unsigned long tell() const
00709 {
00710 unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
00711 w -= _current_item * 2;
00712 _tag = l4_msgtag(0, w, _current_item, 0);
00713 }
00714 #endif
00715 public:
00716 l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00717 {
00718 unsigned w = (_pos + sizeof(l4_umword_t) - 1) / sizeof(
00719 l4_umword_t);
00720 w -= _current_item * 2;
00721 return l4_msgtag(proto, w, _current_item, flags);
00722 }
00723
00724 // XXX: this is a hack for <l4/sys/cxx/ipc_server> adaption
00725 void set_ipc_params(l4_msgtag_t tag)
00726 {
00727 _pos = (tag.words() + tag.items() * 2) * sizeof(l4_umword_t);
00728 _current_item = tag.items();
00729 }
00730 protected:
00731 l4_msgtag_t _tag;
00732 l4_utcb_t *_utcb;
00733 char *_current_msg;
00734 unsigned _pos;
00735 unsigned char _current_item;
00736 };
00737
00738 class Iostream : public Istream, public Ostream
00739 {
00740 public:
00741 explicit Iostream(l4_utcb_t *utcb)
00742 : Istream(utcb), Ostream(utcb)
00743 {}
00744
00745 // disambiguate those functions
00746 l4_msgtag_t tag() const { return Istream::tag(); }

```



```

00811 l4_msgtag_t &tag() { return Istream::tag(); }
00812 l4_utcb_t *utcb() const { return Istream::utcb(); }
00813
00819 void reset()
00820 {
00821 Istream::reset();
00822 Ostream::reset();
00823 }
00824
00825
00833
00834 using Istream::get;
00835 using Istream::put;
00836 using Ostream::put;
00837
00839
00844
00860 inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long
proto = 0);
00861 inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00862
00878 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto =
0)
00879 { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00880
00881 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest,
l4_umword_t *src,
00882 long proto = 0)
00883 { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00884
00901 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
l4_timeout_t timeout, long proto = 0);
00902 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest,
l4_umword_t *src,
00903 l4_timeout_t timeout, long proto = 0);
00904 inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00905 inline l4_msgtag_t reply(long proto = 0)
00906 { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }
00907
00908
00910 };
00911
00912
00913 inline bool
00914 Ostream::put_snd_item(Snd_item const &v)
00915 {
00916 typedef Snd_item T;
00917 _pos = cxx::Type_traits<Snd_item>::align(_pos);
00918 if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00919 return false;
00920
00921 *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00922 _pos += sizeof(T);
00923 ++_current_item;
00924 return true;
00925 }
00926
00927
00928 inline bool
00929 Istream::put(Buf_item const &item)
00930 {
00931 if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 3)
00932 return false;
00933
00934 l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00935
00936 reinterpret_cast<Buf_item&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00937 _current_buf += 2;
00938 return true;
00939 }
00940
00941
00942 inline bool
00943 Istream::put(Small_buf const &item)
00944 {
00945 if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00946 return false;
00947
00948 l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00949
00950 reinterpret_cast<Small_buf&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00951 _current_buf += 1;
00952 return true;
00953 }
00954
00955
00956 inline l4_msgtag_t
00957 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)

```

```

00958 {
00959 l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00960 return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
00961 }
00962
00963 inline l4_msgtag_t
00964 Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00965 {
00966 l4_msgtag_t tag = prepare_ipc(label);
00967 tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
00968 Istream::tag() = tag;
00969 Istream::_pos = 0;
00970 return tag;
00971 }
00972
00973 inline l4_msgtag_t
00974 Iostream::call(l4_cap_idx_t dst, long label)
00975 { return call(dst, L4_IPC_NEVER, label); }
00976
00977
00978 inline l4_msgtag_t
00979 Iostream::reply_and_wait(l4_umword_t *src_dst,
00980 l4_timeout_t timeout, long proto)
00981 {
00982 l4_msgtag_t tag = prepare_ipc(proto);
00983 tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
00984 Istream::tag() = tag;
00985 Istream::_pos = 0;
00986 return tag;
00987 }
00988
00989 inline l4_msgtag_t
00990 Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00991 l4_timeout_t timeout, long proto)
00992 {
00993 l4_msgtag_t tag = prepare_ipc(proto);
00994 tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
00995 Istream::tag() = tag;
00996 Istream::_pos = 0;
00997 return tag;
00998 }
00999
01000 inline l4_msgtag_t
01001 Iostream::reply(l4_timeout_t timeout, long proto)
01002 {
01003 l4_msgtag_t tag = prepare_ipc(proto);
01004 tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, Ostream::_utcb,
01005 tag, timeout);
01006 Istream::tag() = tag;
01007 Istream::_pos = 0;
01008 return tag;
01009 }
01010
01011 inline l4_msgtag_t
01012 Istream::wait(l4_umword_t *src, l4_timeout_t timeout)
01013 {
01014 l4_msgtag_t res;
01015 res = l4_ipc_wait(_utcb, src, timeout);
01016 tag() = res;
01017 _pos = 0;
01018 return res;
01019 }
01020
01021 inline l4_msgtag_t
01022 Istream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
01023 {
01024 l4_msgtag_t res;
01025 res = l4_ipc_receive(src, _utcb, timeout);
01026 tag() = res;
01027 _pos = 0;
01028 return res;
01029 }
01030
01031 } // namespace Ipc
01032 } // namespace L4
01033
01042 inline L4::Ipc::Istream &operator >> (
01043 L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
01044 inline L4::Ipc::Istream &operator >> (
01045 L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
01046 inline L4::Ipc::Istream &operator >> (
01047 L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
01048 inline L4::Ipc::Istream &operator >> (
01049 L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
01050 inline L4::Ipc::Istream &operator >> (

```

```

 L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
01047 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }
01048 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v); return s; }
01049 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
01050 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
01051 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
01052 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
01053 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
01054 inline L4::Ipc::Istream &operator << (
 L4::Ipc::Istream &s, L4::Ipc::Buf_item const &v) { s.put(v); return s; }
01055 inline L4::Ipc::Istream &operator << (
 L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v); return s; }
01056 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, L4::Ipc::Snd_item &v)
01057 {
01058 l4_umword_t b, d;
01059 s >> b >> d;
01060 v = L4::Ipc::Snd_item(b, d);
01061 return s;
01062 }
01063 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, L4::Ipc::Varg &v)
01064 { s.get(&v); return s; }
01065
01066
01075 inline
01076 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4_msgtag_t &v)
01077 {
01078 v = s.tag();
01079 return s;
01080 }
01081
01099 template< typename T >
01100 inline
01101 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Internal::Buf_in<T> const &v)
01102 {
01103 {
01104 unsigned long si;
01105 if (s.get(si) && s.has_more<T>(si))
01106 v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
01107 else
01108 v.set_size(0);
01109 return s;
01110 }
01111
01126 template< typename T >
01127 inline
01128 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Msg_ptr<T> const &v)
01129 {
01130 {
01131 s.get(v);
01132 return s;
01133 }
01134
01147 template< typename T >
01148 inline
01149 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Internal::Buf_cp_in<T> const &v)
01150 {
01151 {
01152 unsigned long sz;
01153 s.get(sz);
01154 v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
01155 return s;
01156 }
01157
01168 template< typename T >
01169 inline
01170 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Str_cp_in<T> const &v)
01171 {
01172 {
01173 unsigned long sz;
01174 s.get(sz);
01175 unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
01176 if (rsz < v.size() && v.buf()[rsz - 1])
01177 ++rsz; // add the zero termination behind the received data
01178
01179 if (rsz != 0)
01180 v.buf()[rsz - 1] = 0;
01181

```

```

01182 v.size() = rsz;
01183 return s;
01184 }
01185
01186
01195 inline L4::Ipc::Ostream &operator << (
01196 L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
01197 inline L4::Ipc::Ostream &operator << (
01198 L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
01199 inline L4::Ipc::Ostream &operator << (
01200 L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
01201 inline L4::Ipc::Ostream &operator << (
01202 L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
01203 inline L4::Ipc::Ostream &operator << (
01204 L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
01205 inline L4::Ipc::Ostream &operator << (
01206 L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
01207 inline L4::Ipc::Ostream &operator << (
01208 L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return s; }
01209 inline L4::Ipc::Ostream &operator << (
01210 L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
01211 inline L4::Ipc::Ostream &operator << (
01212 L4::Ipc::Ostream &s, signed short int v) { s.put(v); return s; }
01213 inline L4::Ipc::Ostream &operator << (
01214 L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s; }
01215 inline L4::Ipc::Ostream &operator << (
01216 L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
01217 inline L4::Ipc::Ostream &operator << (
01218 L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
01219 inline L4::Ipc::Ostream &operator << (
01220 L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
01221 inline L4::Ipc::Ostream &operator << (
01222 L4::Ipc::Ostream &s, L4::Ipc::Snd_item const &v) { s.put_snd_item(v);
01223 return s; }
01224 template< typename T >
01225 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
01226 { s << L4::Ipc::Snd_fpage(v.fpage()); return s; }
01227
01228 inline L4::Ipc::Ostream &operator << (
01229 L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
01230 { s.put(v); return s; }
01231 template< typename T >
01232 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
01233 { s.put(v); return s; }
01234
01235 inline
01236 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s,
01237 L4_msgtag_t const &v)
01238 {
01239 s.tag() = v;
01240 return s;
01241 }
01242
01243 template< typename T >
01244 inline
01245 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s,
01246 L4::Ipc::Internal::Buf_cp_out<T> const &v)
01247 {
01248 s.put(v.size());
01249 s.put(v.buf(), v.size());
01250 return s;
01251 }
01252
01253 inline
01254 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, char const
01255 *v)
01256 {
01257 unsigned long l = __builtin_strlen(v) + 1;
01258 s.put(l);
01259 s.put(v, l);
01260 return s;
01261 }
01262
01263 #ifdef L4_CXX_IPC_BACKWARD_COMPAT
01264 namespace L4 {
01265
01266 #if 0
01267 template< typename T > class Ipc_buf_cp_out : public Ipc::Buf_cp_out<T> {};
01268 template< typename T > class Ipc_buf_cp_in : public Ipc::Buf_cp_in<T> {};
01269 template< typename T > class Ipc_buf_in : public Ipc::Buf_in<T> {};
01270 template< typename T > class Msg_ptr : public Ipc::Msg_ptr<T> {};
01271 #endif
01272
01273 template< typename T >
01274 Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
01275 L4_DEPRECATED("Use L4::Ipc::buf_cp_out() now");
01276
01277 template< typename T >

```

```

01291 Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
01292 { return Ipc::Internal::Buf_cp_out<T>(v, size); }
01293
01294
01295 template< typename T >
01296 Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
01297 L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");
01298
01299 template< typename T >
01300 Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
01301 { return Ipc::Internal::Buf_cp_in<T>(v, size); }
01302
01303
01304 template< typename T >
01305 Ipc::Internal::Buf_in<T> ipc_buf_in(T *v, unsigned long &size)
01306 L4_DEPRECATED("Use L4::Ipc::buf_in() now");
01307
01308 template< typename T >
01309 Ipc::Internal::Buf_in<T> ipc_buf_in(T *v, unsigned long &size)
01310 { return Ipc::Internal::Buf_in<T>(v, size); }
01311
01312
01313 template< typename T >
01314 Ipc::Msg_ptr<T> msg_ptr(T *p)
01315 L4_DEPRECATED("Use L4::Ipc::msg_ptr() now");
01316
01317 template< typename T >
01318 Ipc::Msg_ptr<T> msg_ptr(T *p)
01319 { return Ipc::Msg_ptr<T>(p); }
01320
01321 typedef Ipc::Istream Ipc_istream L4_DEPRECATED("Use L4::Ipc::Istream now");
01322 typedef Ipc::Ostream Ipc_ostream L4_DEPRECATED("Use L4::Ipc::Ostream now");
01323 typedef Ipc::Iostream Ipc_iostream L4_DEPRECATED("Use L4::Ipc::Iostream now");
01324 typedef Ipc::Snd_fpage Snd_fpage L4_DEPRECATED("Use L4::Ipc::Snd_fpage
now");
01325 typedef Ipc::Rcv_fpage Rcv_fpage L4_DEPRECATED("Use L4::Ipc::Rcv_fpage
now");
01326 typedef Ipc::Small_buf Small_buf L4_DEPRECATED("Use L4::Ipc::Small_buf now");
01327
01328
01329 namespace Ipc {
01330 template< typename T > class Buf_cp_in : public Internal::Buf_cp_in<T>
01331 {
01332 public:
01333 Buf_cp_in(T *v, unsigned long &size) : Internal::Buf_cp_in<T>(v, size) {}
01334 };
01335
01336 template< typename T >
01337 class Buf_cp_out : public Internal::Buf_cp_out<T>
01338 {
01339 public:
01340 Buf_cp_out(T const *v, unsigned long size) : Internal::Buf_cp_out<T>(v, size) {}
01341 };
01342
01343 template< typename T >
01344 class Buf_in : public Internal::Buf_in<T>
01345 {
01346 public:
01347 Buf_in(T *v, unsigned long &size) : Internal::Buf_in<T>(v, size) {}
01348 };
01349 } // namespace Ipc
01350 } // namespace L4
01351
01352 template< typename T >
01353 inline
01354 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_cp_in<T> const &v)
01355 L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");
01356
01357 template< typename T >
01358 inline
01359 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_cp_in<T> const &v)
01360 { return operator>>(s, static_cast<L4::Ipc::Internal::Buf_cp_in<T> >(v)); }
01361
01362 template< typename T >
01363 inline
01364 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_in<T> const &v)
01365 L4_DEPRECATED("Use L4::Ipc::buf_in() now");
01366
01367 template< typename T >
01368 inline
01369 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
L4::Ipc::Buf_in<T> const &v)
01370 { return operator>>(s, static_cast<L4::Ipc::Internal::Buf_in<T> >(v)); }
01371

```



## 15.118 l4iostream

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/cxx/basic_ostream>
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/capability>
00029
00030 inline
00031 L4::BasicOStream &operator << (L4::BasicOStream &o, l4_msgtag_t const &tag)
00032 {
00033 L4::IOBackend::Mode m = o.be_mode();
00034 o << "[l=" << L4::dec << tag.label() << "; w=" << tag.words() << "; i="
00035 << tag.items() << "];";
00036 o.be_mode(m);
00037 return o;
00038 }
00039
00040 template<typename T>
00041 inline
00042 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::Cap<T> const &cap)
00043 {
00044 o << "[C:" << L4::n_hex(cap.cap()) << "];";
00045 return o;
00046 }

```

## 15.119 l4/cxx/l4types.h File Reference

[L4 Types](#).

```

#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>

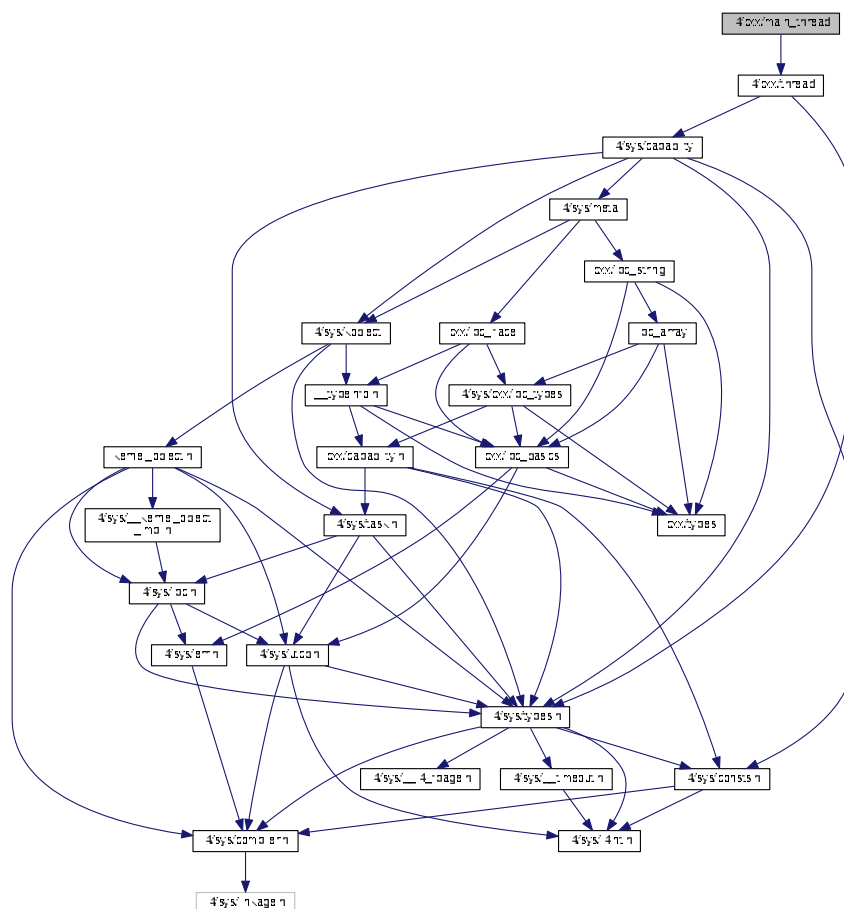
```





### 15.121 I4/cxx/main thread File Reference

Include dependency graph for main thread:



## Namespaces

- [CXX](#)

*Our C++ library.*

### 15.121.1 Detailed Description

Main thread.

Definition in file [main\\_thread](#).

### 15.122 main\_thread

```

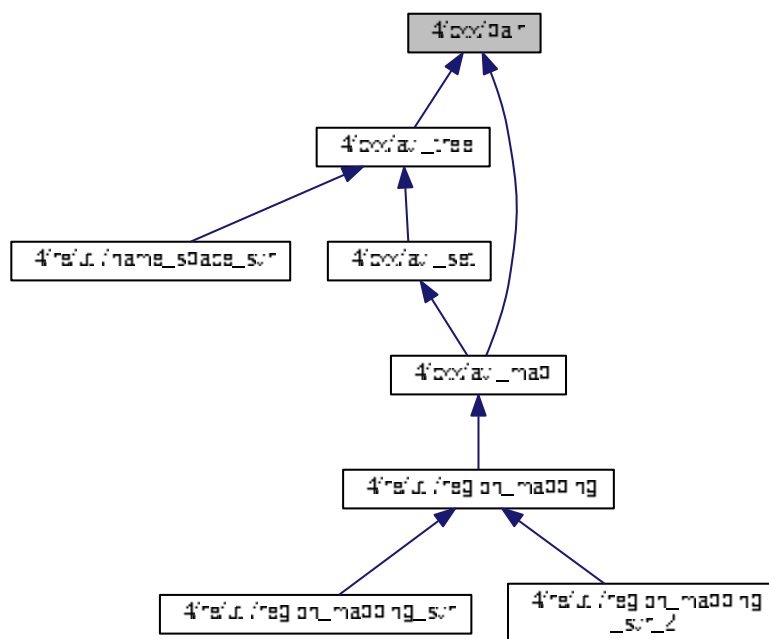
00001
00005 /*
00006 * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023
00024 #ifndef L4_CXX_MAIN_THREAD_H__
00025 #define L4_CXX_MAIN_THREAD_H__
00026
00027 #include <l4/cxx/thread>
00028
00029 namespace cxx {
00030 class MainThread : public Thread
00031 {
00032 public:
00033 MainThread() : Thread(true)
00034 {}
00035 };
00036 };
00037
00038 #endif /* L4_CXX_MAIN_THREAD_H__ */

```

### 15.123 l4/cxx/pair File Reference

Pair implementation.

This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `cxx::Pair< First, Second >`  
*Pair of two values.*
- class `cxx::Pair_first_compare< Cmp, Typ >`  
*Comparison functor for `Pair`.*

## Namespaces

- CXX
- Our C++ library.*

### 15.123.1 Detailed Description

Pair implementation.

Definition in file [pair](#).

## 15.124 pair

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace cxx {
00026
00035 template< typename First, typename Second >
00036 struct Pair
00037 {
00039 typedef First First_type;
00041 typedef Second Second_type;
00042
00044 First first;
00046 Second second;
00047
00053 template<typename A1, typename A2>
00054 Pair(A1 &&first, A2 &&second)
00055 : first(first), second(second) {}
00056
00058 Pair() {}
00059 };
00060
00061 template< typename F, typename S >
00062 Pair<F,S> pair(F const &f, S const &s)
00063 { return cxx::Pair<F,S>(f,s); }
00064
00065
00074 template< typename Cmp, typename Typ >
00075 class Pair_first_compare
00076 {
00077 private:
00078 Cmp const &_cmp;
00079
00080 public:
00085 Pair_first_compare(Cmp const &cmp = Cmp()) : _cmp(cmp) {}
00086
00092 bool operator () (Typ const &l, Typ const &r) const
00093 { return _cmp(l.first,r.first); }
00094 };
00095
00096 }
00097
00098 template< typename OS, typename A, typename B >
00099 inline
00100 OS &operator << (OS &os, cxx::Pair<A,B> const &p)
00101 {
00102 os << p.first << ';<' << p.second;
00103 return os;
00104 }
00105

```

## 15.125 l4/cxx/std\_exc\_io File Reference

Base exceptions std stream operator.

```

#include <l4/cxx/exceptions>
#include <iostream>

```

The graph illustrates the relationships between various mathematical concepts, organized into a hierarchical structure. The nodes are labeled with mathematical terms, and the edges represent relationships between them. The graph is highly interconnected, with many nodes having multiple incoming and outgoing edges.

The nodes are labeled with mathematical terms, and the edges represent relationships between them. The graph is highly interconnected, with many nodes having multiple incoming and outgoing edges.

Definition in file [std\\_exc\\_io](#).

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021
00022 #include <l4/cxx/exceptions>
00023 #include <iostream>
00024
00025 inline
```

```

00031 std::ostream &
00032 operator << (std::ostream &o, L4::Base_exception const &e)
00033 {
00034 o << "Exception: " << e.str() << ", backtrace ...\n";
00035 for (int i = 0; i < e.frame_count(); ++i)
00036 o << (void *) (e.pc_array()[i]) << '\n';
00037 return o;
00038 }
00039
00040
00041 inline
00042 std::ostream &
00043 operator << (std::ostream &o, L4::Runtime_error const &e)
00044 {
00045 o << "Exception: " << e.str() << ": ";
00046 if (e.extra_str())
00047 o << e.extra_str() << ": ";
00048 o << "backtrace ...\n";
00049 for (int i = 0; i < e.frame_count(); ++i)
00050 o << (void *) (e.pc_array()[i]) << '\n';
00051 return o;
00052 }
00053 }

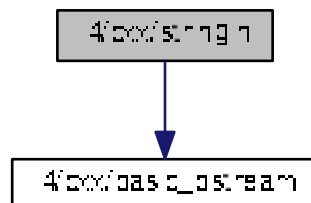
```

## 15.127 l4/cxx/string.h File Reference

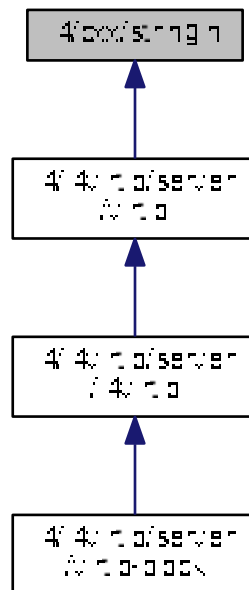
String.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for string.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::String](#)

*A null-terminated string container class.*

## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

## 15.127.1 Detailed Description

String.

Definition in file [string.h](#).

## 15.128 string.h

```

00001
00005 /*
00006 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/cxx/basic_ostream>
00026
00027 namespace L4 {
00028
00033 class String
00034 {
00035 public:
00036 String(char const *str = "") : _str(str)
00037 {}
00038
00039 unsigned length() const
00040 {
00041 unsigned l;
00042 for (l = 0; _str[l]; l++)
00043 ;
00044 return l;
00045 }
00046
00047 char const *p_str() const { return _str; }
00048
00049 private:
00050 char const *_str;
00051 };
00052 }
00053
00054 inline
00055 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::String const &s)
00056 {
00057 o << s.p_str();
00058 return o;
00059 }

```

## 15.129 l4/irq/irq.h File Reference

IRQ handling routines.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```



```

graph TD
 A[ר] --> B[ר]
 A --> C[ר]
 A --> D[ר]
 B --> E[ר]
 B --> F[ר]
 B --> G[ר]
 C --> D
 C --> E
 C --> F
 D --> E
 D --> F
 E --> G
 F --> G
 G --> H[ר]

```

- `l4irq_t * l4irq_attach` (int irqnum)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_ft` (int irqnum, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread` (int irqnum, `l4_cap_idx_t` to\_thread)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_thread_ft` (int irqnum, `l4_cap_idx_t` to\_thread, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `long l4irq_wait` (`l4irq_t *irq`)  
*Wait for specified IRQ.*
- `long l4irq_unmask_and_wait_any` (`l4irq_t *unmask_irq`, `l4irq_t **ret_irq`)  
*Unmask a specific IRQ and wait for any attached IRQ.*
- `long l4irq_wait_any` (`l4irq_t **irq`)  
*Wait for any attached IRQ.*
- `long l4irq_unmask` (`l4irq_t *irq`)  
*Unmask a specific IRQ.*
- `long l4irq_detach` (`l4irq_t *irq`)  
*Detach from IRQ.*
- `l4irq_t * l4irq_request` (int irqnum, void(\*isr\_handler)(void \*), void \*isr\_data, int irq\_thread\_prio, unsigned mode)  
*Attach asynchronous ISR handler to IRQ.*
- `long l4irq_release` (`l4irq_t *irq`)  
*Release asynchronous ISR handler and free resources.*
- `l4irq_t * l4irq_attach_cap` (`l4_cap_idx_t` irqcap)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_cap_ft` (`l4_cap_idx_t` irqcap, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread_cap` (`l4_cap_idx_t` irqcap, `l4_cap_idx_t` to\_thread)  
*Attach/connect to IRQ.*

- `l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)`  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_request_cap (l4_cap_idx_t irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_↵prio, unsigned mode)`  
*Attach asynchronous ISR handler to IRQ.*

### 15.129.1 Detailed Description

IRQ handling routines.

Definition in file [irq.h](#).

## 15.130 irq.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Henning Schild <hschild@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 */
00014 #pragma once
00015
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/types.h>
00018
00019 __BEGIN_DECLS
00020
00028 struct l4irq_t;
00029 typedef struct l4irq_t l4irq_t;
00030
00041 L4_CV l4irq_t *
00042 l4irq_attach(int irqnum);
00043
00055 L4_CV l4irq_t *
00056 l4irq_attach_ft(int irqnum, unsigned mode);
00057
00068 L4_CV l4irq_t *
00069 l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);
00070
00082 L4_CV l4irq_t *
00083 l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
00084 unsigned mode);
00085
00093 L4_CV long
00094 l4irq_wait(l4irq_t *irq);
00095
00104 L4_CV long
00105 l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00106
00114 L4_CV long
00115 l4irq_wait_any(l4irq_t **irq);
00116
00127 L4_CV long
00128 l4irq_unmask(l4irq_t *irq);
00129
00137 L4_CV long
00138 l4irq_detach(l4irq_t *irq);
00139
00140
00141
00142 /*****
00164 L4_CV l4irq_t *
00165 l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00166 int irq_thread_prio, unsigned mode);
00167
00175 L4_CV long
00176 l4irq_release(l4irq_t *irq);
00177
00178

```

```

00179
00180 /*****
00181 /*****
00182
00188 L4_CV l4irq_t *
00199 l4irq_attach_cap(l4_cap_idx_t irqcap);
00200
00212 L4_CV l4irq_t *
00213 l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);
00214
00225 L4_CV l4irq_t *
00226 l4irq_attach_thread_cap(l4_cap_idx_t irqcap,
00227 l4_cap_idx_t to_thread);
00227
00239 L4_CV l4irq_t *
00240 l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap,
00241 l4_cap_idx_t to_thread,
00242 unsigned mode);
00242
00243 /*****
00264 L4_CV l4irq_t *
00265 l4irq_request_cap(l4_cap_idx_t irqcap,
00266 void (*isr_handler)(void *), void *isr_data,
00267 int irq_thread_prio, unsigned mode);
00268
00269 __END_DECLS

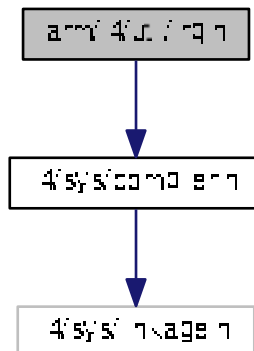
```

## 15.131 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for irq.h:



### 15.131.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file [irq.h](#).

## 15.132 irq.h

```

00001
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00017 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00018
00019 #ifdef __GNUC__
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_INLINE void l4util_cli (void);
00026 L4_INLINE void l4util_sti (void);
00027 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00028 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00029
00030 L4_INLINE
00031 void
00032 l4util_cli(void)
00033 {
00034 extern void __do_not_use_l4util_cli(void);
00035 __do_not_use_l4util_cli();
00036 }
00037
00038
00039 L4_INLINE
00040 void
00041 l4util_sti(void)
00042 {
00043 extern void __do_not_use_l4util_sti(void);
00044 __do_not_use_l4util_sti();
00045 }
00046
00047
00048 L4_INLINE
00049 void
00050 l4util_flags_save(l4_umword_t *flags)
00051 {
00052 (void) flags;
00053 extern void __do_not_use_l4util_flags_save(void);
00054 __do_not_use_l4util_flags_save();
00055 }
00056
00057 L4_INLINE
00058 void
00059 l4util_flags_restore(l4_umword_t *flags)
00060 {
00061 (void) flags;
00062 extern void __do_not_use_l4util_flags_restore(void);
00063 __do_not_use_l4util_flags_restore();
00064 }
00065
00066 EXTERN_C_END
00067
00068 #endif // __GNUC__
00069
00070 #endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */

```

## 15.133 amd64/l4/util/irq.h File Reference

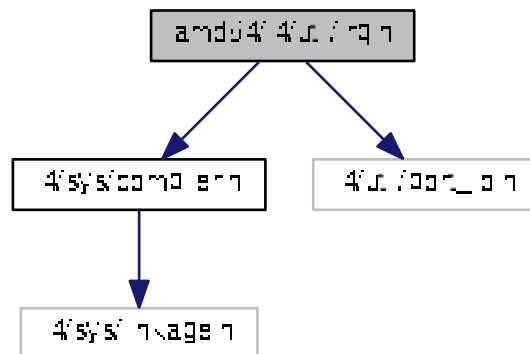
some PIC and hardware interrupt related functions

```

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>

```

Include dependency graph for irq.h:



## Functions

- void [l4util\\_irq\\_acknowledge](#) (unsigned int irq)  
*Acknowledge IRQ at PIC in fully special nested mode.*

### 15.133.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [irq.h](#).

### 15.133.2 Function Documentation

#### 15.133.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (
 unsigned int irq) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

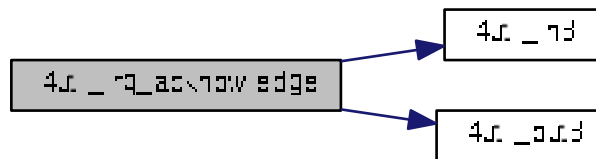
## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>irq</i> | number of interrupt to acknowledge |
|------------|------------------------------------|

Definition at line 66 of file [irq.h](#).

References [EXTERN\\_C\\_END](#), [l4util\\_in8\(\)](#), and [l4util\\_out8\(\)](#).

Here is the call graph for this function:



## 15.134 irq.h

```

00001
00010 /*
00011 * (c) 2003-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00029 L4_INLINE void
00030 l4util_irq_acknowledge(unsigned int irq);
00031
00034 static inline void
00035 l4util_cli(void)
00036 {
00037 __asm__ __volatile__ ("cli" : : : "memory");
00038 }
00039
00042 static inline void
00043 l4util_sti(void)
00044 {
00045 __asm__ __volatile__ ("sti" : : : "memory");
00046 }
00047
00051 static inline void
00052 l4util_flags_save(l4_umword_t *flags)
00053 {
00054 __asm__ __volatile__ ("pushf ; popq %0" : "=g" (*flags) : : "memory");
00055 }
00056
00059 static inline void
00060 l4util_flags_restore(l4_umword_t *flags)
00061 {
00062 __asm__ __volatile__ ("pushq %0 ; popf" : : "g" (*flags) : "memory");
00063 }

```

```

00064
00065 L4_INLINE void
00066 l4util_irq_acknowledge(unsigned int irq)
00067 {
00068 if (irq > 7)
00069 {
00070 l4util_out8(0x60+(irq & 7), 0xA0);
00071 l4util_out8(0x0B, 0xA0);
00072 if (l4util_in8(0xA0) == 0)
00073 l4util_out8(0x60 + 2, 0x20);
00074 }
00075 else
00076 l4util_out8(0x60+irq, 0x20); /* acknowledge the irq */
00077 };
00078
00079 EXTERN_C_END
00080
00081 #endif

```

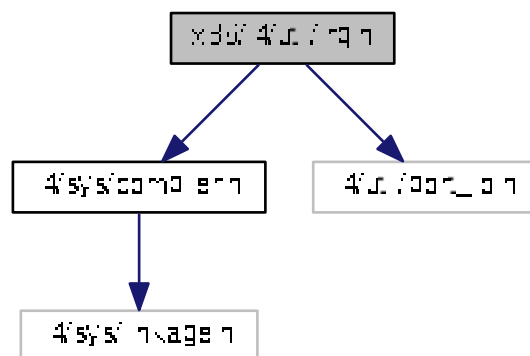
## 15.135 x86/I4/util/irq.h File Reference

some PIC and hardware interrupt related functions

```

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>
Include dependency graph for irq.h:

```



### Functions

- void `l4util_irq_acknowledge` (unsigned int irq)  
Acknowledge IRQ at PIC in fully special nested mode.

### 15.135.1 Detailed Description

some PIC and hardware interrupt related functions

## Date

2003

## Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)Definition in file [irq.h](#).

## 15.135.2 Function Documentation

## 15.135.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (
 unsigned int irq) [inline]
```

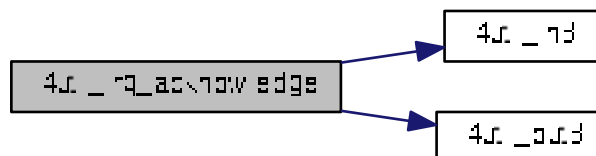
Acknowledge IRQ at PIC in fully special nested mode.

## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>irq</i> | number of interrupt to acknowledge |
|------------|------------------------------------|

Definition at line 66 of file [irq.h](#).References [EXTERN\\_C\\_END](#), [l4util\\_in8\(\)](#), and [l4util\\_out8\(\)](#).

Here is the call graph for this function:



## 15.136 irq.h

```
00001
00010 /*
00011 * (c) 2003-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
```



```

00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00026 L4_INLINE void
00027 l4util_irq_acknowledge(unsigned int irq);
00028
00029 static inline void
00030 l4util_cli (void)
00031 {
00032 __asm__ __volatile__ ("cli" : : : "memory");
00033 }
00034
00035 static inline void
00036 l4util_sti (void)
00037 {
00038 __asm__ __volatile__ ("sti" : : : "memory");
00039 }
00040
00041 static inline void
00042 l4util_flags_save (l4_umword_t *flags)
00043 {
00044 __asm__ __volatile__ ("pushfl ; popl %0" : "=g" (*flags) : : "memory");
00045 }
00046
00047 static inline void
00048 l4util_flags_restore (l4_umword_t *flags)
00049 {
00050 __asm__ __volatile__ ("pushl %0 ; popfl" : : "g" (*flags) : "memory");
00051 }
00052
00053 L4_INLINE void
00054 l4util_irq_acknowledge(unsigned int irq)
00055 {
00056 if (irq > 7)
00057 {
00058 l4util_out8(0x60+(irq & 7), 0xA0);
00059 l4util_out8(0x0B, 0xA0);
00060 if (l4util_in8(0xA0) == 0)
00061 l4util_out8(0x60 + 2, 0x20);
00062 }
00063 else
00064 l4util_out8(0x60+irq, 0x20); /* acknowledge the irq */
00065 };
00066
00067 EXTERN_C_END
00068
00069 #endif

```

## 15.137 l4/sys/irq.h File Reference

C Irq interface.

```

#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
#include <l4/sys/rcv_endpoint.h>

```

- `l4_msgtag_t l4_irq_attach (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread) L4_NOTHROW`  
*Attach a thread to an interrupt source.*
- `l4_msgtag_t l4_irq_attach_u (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW`  
*Attach a thread to this interrupt.*
- `l4_msgtag_t l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW`  
*Chain an IRQ to another master IRQ source.*
- `l4_msgtag_t l4_irq_mux_chain_u (l4_cap_idx_t irq, l4_cap_idx_t slave, l4_utcb_t *utcb) L4_NOTHROW`

- Attach an IRQ to this multiplexer.*
- [l4\\_msgtag\\_t l4\\_irq\\_detach \(l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)
- Detach from an interrupt source.*
- [l4\\_msgtag\\_t l4\\_irq\\_detach\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)
- Detach from this interrupt.*
- [l4\\_msgtag\\_t l4\\_irq\\_trigger \(l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)
- Trigger an IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_trigger\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)
- Trigger.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive \(l4\\_cap\\_idx\\_t irq, l4\\_timeout\\_t to\) L4\\_NOTHROW](#)
- Unmask and wait for specified IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_timeout\\_t timeout, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)
- Unmask and wait for this IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait \(l4\\_cap\\_idx\\_t irq, l4\\_umword\\_t \\*label, l4\\_timeout\\_t to\) L4\\_NOTHROW](#)
- Unmask IRQ and wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_umword\\_t \\*label, l4\\_timeout\\_t timeout, l4\\_utcb\\_t \\*utcb\) L4↔\\_NOTHROW](#)
- Unmask IRQ and (open) wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask \(l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)
- Unmask IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)
- Unmask IRQ.*

### 15.137.1 Detailed Description

C Irq interface.

Definition in file [irq.h](#).

## 15.138 irq.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029 #include <l4/sys/rcv_endpoint.h>
00030
00066 L4_INLINE l4_msgtag_t
00067 l4_irq_attach(l4_cap_idx_t irq, l4_umword_t label,
00068 l4_cap_idx_t thread) L4_NOTHROW
00069 L4_DEPRECATED("Use l4_rcv_ep_bind_thread().");

```

```

00070
00078 L4_INLINE l4_msgtag_t
00079 l4_irq_attach_u(l4_cap_idx_t irq, l4_umword_t label,
00080 l4_cap_idx_t thread, l4_utcb_t *utcb)
00081 L4_NOTHROW
00082 L4_DEPRECATED("Use l4_rcv_ep_bind_thread_u().");
00083
00102 L4_INLINE l4_msgtag_t
00103 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave)
00104 L4_NOTHROW;
00105
00111 L4_INLINE l4_msgtag_t
00112 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00113 l4_utcb_t *utcb) L4_NOTHROW;
00114
00123 L4_INLINE l4_msgtag_t
00124 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;
00125
00132 L4_INLINE l4_msgtag_t
00133 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
00134 L4_NOTHROW;
00135
00151 L4_INLINE l4_msgtag_t
00152 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;
00153
00160 L4_INLINE l4_msgtag_t
00161 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
00162 L4_NOTHROW;
00163
00172 L4_INLINE l4_msgtag_t
00173 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to)
00174 L4_NOTHROW;
00175
00181 L4_INLINE l4_msgtag_t
00182 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout,
00183 l4_utcb_t *utcb) L4_NOTHROW;
00184
00194 L4_INLINE l4_msgtag_t
00195 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00196 l4_timeout_t to) L4_NOTHROW;
00197
00204 L4_INLINE l4_msgtag_t
00205 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00206 l4_timeout_t timeout, l4_utcb_t *utcb)
00207 L4_NOTHROW;
00208
00218 L4_INLINE l4_msgtag_t
00219 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;
00220
00227 L4_INLINE l4_msgtag_t
00228 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
00229 L4_NOTHROW;
00230
00233 enum L4_irq_sender_op
00234 {
00235 L4_IRQ_SENDER_OP_ATTACH = 0,
00236 L4_IRQ_SENDER_OP_DETACH = 1
00237 };
00238
00242 enum L4_irq_mux_op
00243 {
00244 L4_IRQ_MUX_OP_CHAIN = 0
00245 };
00246
00250 enum L4_irq_op
00251 {
00252 L4_IRQ_OP_TRIGGER = 2,
00253 L4_IRQ_OP_EOI = 4
00254 };
00255
00256 /*****
00257 * Implementations
00258 */
00259
00260 L4_INLINE l4_msgtag_t
00261 l4_irq_attach_u(l4_cap_idx_t irq, l4_umword_t label,
00262 l4_cap_idx_t thread, l4_utcb_t *utcb)
00263 L4_NOTHROW
00264 {
00265 int items = 0;
00266 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00267 m->mr[0] = L4_IRQ_SENDER_OP_ATTACH;
00268 m->mr[1] = label;
00269 if (!l4_is_invalid_cap(thread))
00270 {

```

```

00271 items = 1;
00272 m->mr[2] = l4_map_obj_control(0, 0);
00273 m->mr[3] = l4_obj_fpage(thread, 0, L4_FPAGE_RWX).
raw;
00274 }
00275 return l4_ipc_call(irq, utcb, l4_msgtag(
L4_PROTO_IRQ_SENDER, 2, items, 0),
L4_IPC_NEVER);
00276 }
00277 }
00278
00279 L4_INLINE l4_msgtag_t
00280 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00281 l4_utcb_t *utcb) L4_NOTHROW
00282 {
00283 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00284 m->mr[0] = L4_IRQ_MUX_OP_CHAIN;
00285 m->mr[1] = l4_map_obj_control(0, 0);
00286 m->mr[2] = l4_obj_fpage(slave, 0, L4_FPAGE_RWX).raw;
00287 return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_MUX, 1, 1, 0),
00288 L4_IPC_NEVER);
00289 }
00290
00291 L4_INLINE l4_msgtag_t
00292 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
L4_NOTHROW
00293 {
00294 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_SENDER_OP_DETACH;
00295 return l4_ipc_call(irq, utcb, l4_msgtag(
L4_PROTO_IRQ_SENDER, 1, 0, 0),
L4_IPC_NEVER);
00296 }
00297 }
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
L4_NOTHROW
00301 {
00302 return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 0, 0, 0),
00303 L4_IPC_BOTH_TIMEOUT_0);
00304 }
00305
00306 L4_INLINE l4_msgtag_t
00307 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t to,
00308 l4_utcb_t *utcb) L4_NOTHROW
00309 {
00310 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00311 return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), to);
00312 }
00313 L4_INLINE l4_msgtag_t
00314 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00315 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00316 {
00317 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00318 return l4_ipc_send_and_wait(irq, utcb, l4_msgtag(
L4_PROTO_IRQ, 1, 0, 0),
label, to);
00319 }
00320 }
00321
00322 L4_INLINE l4_msgtag_t
00323 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
L4_NOTHROW
00324 {
00325 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00326 return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
L4_IPC_NEVER);
00327 }
00328
00329
00330 L4_INLINE l4_msgtag_t
00331 l4_irq_attach(l4_cap_idx_t irq, l4_umword_t label,
00332 l4_cap_idx_t thread) L4_NOTHROW
00333 {
00334 #pragma GCC diagnostic push
00335 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00336 return l4_irq_attach_u(irq, label, thread, l4_utcb());
00337 #pragma GCC diagnostic pop
00338 }
00339
00340 L4_INLINE l4_msgtag_t
00341 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave)
L4_NOTHROW
00342 {
00343 return l4_irq_mux_chain_u(irq, slave, l4_utcb());
00344 }
00345
00346 L4_INLINE l4_msgtag_t
00347 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW

```

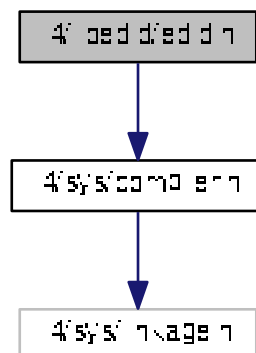
```

00348 {
00349 return l4_irq_detach_u(irq, l4_utcb());
00350 }
00351
00352 L4_INLINE l4_msgtag_t
00353 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW
00354 {
00355 return l4_irq_trigger_u(irq, l4_utcb());
00356 }
00357
00358 L4_INLINE l4_msgtag_t
00359 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to)
 L4_NOTHROW
00360 {
00361 return l4_irq_receive_u(irq, to, l4_utcb());
00362 }
00363
00364 L4_INLINE l4_msgtag_t
00365 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00366 l4_timeout_t to) L4_NOTHROW
00367 {
00368 return l4_irq_wait_u(irq, label, to, l4_utcb());
00369 }
00370
00371 L4_INLINE l4_msgtag_t
00372 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW
00373 {
00374 return l4_irq_unmask_u(irq, l4_utcb());
00375 }
00376

```

## 15.139 l4/libedid/edid.h File Reference

#include <l4/sys/compiler.h>  
 Include dependency graph for edid.h:



## Enumerations

- enum [Libedid\\_consts](#) { [Libedid\\_block\\_size](#) = 128 }

*EDID constants.*

## Functions

- `int libedid_check_header` (const unsigned char \*edid)  
*Check for valid EDID header.*
- `int libedid_checksum` (const unsigned char \*edid)  
*Calculates the EDID checksum.*
- `unsigned libedid_version` (const unsigned char \*edid)  
*Returns the EDID version number.*
- `unsigned libedid_revision` (const unsigned char \*edid)  
*Returns the EDID revision number.*
- `void libedid_pnp_id` (const unsigned char \*edid, unsigned char \*id)  
*Extracts the display's PnP ID.*
- `void libedid_prefered_resolution` (const unsigned char \*edid, unsigned \*w, unsigned \*h)  
*Extract the display's preferred mode.*
- `unsigned libedid_num_ext_blocks` (const unsigned char \*edid)  
*Get the number of EDID extension blocks.*
- `unsigned libedid_dump_standard_timings` (const unsigned char \*edid)  
*Dump the standard timings to stdout.*
- `void libedid_dump` (const unsigned char \*edid)  
*Dump raw EDID data to stdout.*

## 15.140 edid.h

```

00001
00004 /*
00005 * (c) 2014 Matthias Lange <matthias.lange@kernkonzept.com>
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU Lesser General Public License 2.1.
00009 * Please see the COPYING-LGPL-2.1 file for details.
00010 */
00011 #pragma once
00012
00013 #include <14/sys/compiler.h>
00014
00023 enum Libedid_consts
00024 {
00025 Libedid_block_size = 128,
00026 };
00027
00028 __BEGIN_DECLS
00029
00036 int libedid_check_header(const unsigned char *edid);
00037
00044 int libedid_checksum(const unsigned char *edid);
00045
00052 unsigned libedid_version(const unsigned char *edid);
00053
00060 unsigned libedid_revision(const unsigned char *edid);
00061
00068 void libedid_pnp_id(const unsigned char *edid, unsigned char *id);
00069
00077 void libedid_prefered_resolution(const unsigned char *edid,
00078 unsigned *w, unsigned *h);
00079
00086 unsigned libedid_num_ext_blocks(const unsigned char *edid);
00087
00094 unsigned libedid_dump_standard_timings(const unsigned char *edid);
00095
00101 void libedid_dump(const unsigned char *edid);
00102
00105 __END_DECLS

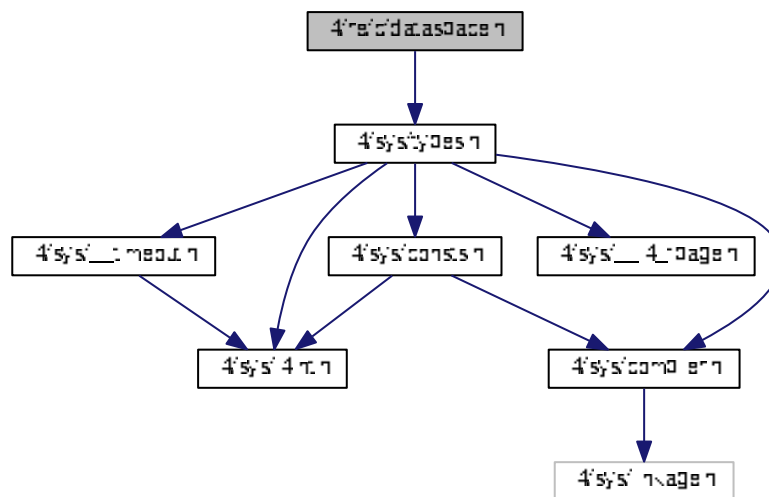
```

## 15.141 l4/re/c/dataspace.h File Reference

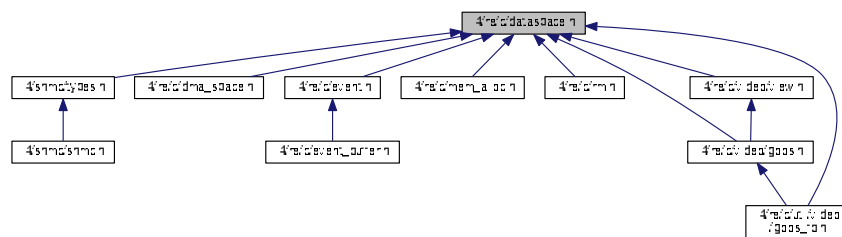
Data space C interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for dataspace.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [l4re\\_ds\\_stats\\_t](#)  
Information about the data space.

### Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_ds\\_t](#)  
Dataspace type.



## Enumerations

- enum `l4re_ds_map_flags` { ,  
`L4RE_DS_MAP_NORMAL` = 0x00, `L4RE_DS_MAP_CACHEABLE` = `L4RE_DS_MAP_NORMAL`, `L4RE_DS_MAP_BUFFERABLE` = 0x10, `L4RE_DS_MAP_UNCACHEABLE` = 0x20,  
`L4RE_DS_MAP_CACHING_MASK` = 0x30, `L4RE_DS_MAP_CACHING_SHIFT` = 4 }

*Flags to specify the memory mapping type of a request.*

## Functions

- long `l4re_ds_clear` (const `l4re_ds_t` ds, `l4_addr_t` offset, unsigned long size) `L4_NOTHROW`
- long `l4re_ds_allocate` (const `l4re_ds_t` ds, `l4_addr_t` offset, `l4_size_t` size) `L4_NOTHROW`
- int `l4re_ds_copy_in` (const `l4re_ds_t` ds, `l4_addr_t` dst\_offs, const `l4re_ds_t` src, `l4_addr_t` src\_offs, unsigned long size) `L4_NOTHROW`
- unsigned long `l4re_ds_size` (const `l4re_ds_t` ds) `L4_NOTHROW`
- long `l4re_ds_flags` (const `l4re_ds_t` ds) `L4_NOTHROW`
- int `l4re_ds_info` (const `l4re_ds_t` ds, `l4re_ds_stats_t` \*stats) `L4_NOTHROW`
- int `l4re_ds_phys` (const `l4re_ds_t` ds, `l4_addr_t` offset, `l4_addr_t` \*phys\_addr, `l4_size_t` \*phys\_size) `L4_NOTHROW`

*Return physical address.*

### 15.141.1 Detailed Description

Data space C interface.

Definition in file [dataspace.h](#).

## 15.142 dataspace.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032
00033 EXTERN_C_BEGIN
00034
00039 typedef l4_cap_idx_t l4re_ds_t;
00040
00045 typedef struct {
00046 unsigned long size;
00047 unsigned long flags;
00048 } l4re_ds_stats_t;
00049
00054 enum l4re_ds_map_flags {
00055 L4RE_DS_MAP_FLAG_RO = 0,

```

```

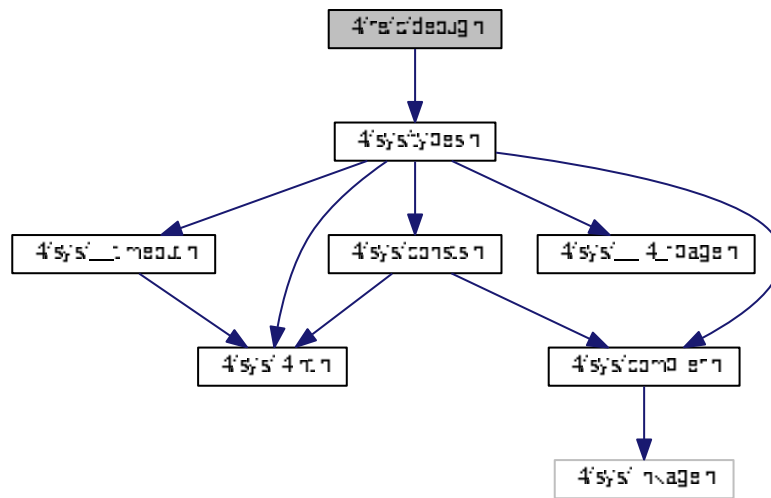
00056 L4RE_DS_MAP_FLAG_RW = 1,
00057
00058 L4RE_DS_MAP_NORMAL = 0x00,
00059 L4RE_DS_MAP_CACHEABLE = L4RE_DS_MAP_NORMAL,
00060 L4RE_DS_MAP_BUFFERABLE = 0x10,
00061 L4RE_DS_MAP_UNCACHEABLE = 0x20,
00062 L4RE_DS_MAP_CACHING_MASK = 0x30,
00063 L4RE_DS_MAP_CACHING_SHIFT = 4,
00064 };
00065
00071 L4_CV int
00072 l4re_ds_map(const l4re_ds_t ds, l4_addr_t offset, unsigned long flags,
00073 l4_addr_t local_addr, l4_addr_t min_addr,
00074 l4_addr_t max_addr) L4_NOTHROW;
00075
00080 L4_CV int
00081 l4re_ds_map_region(const l4re_ds_t ds, l4_addr_t offset, unsigned long flags,
00082 l4_addr_t min_addr, l4_addr_t max_addr)
00083 L4_NOTHROW;
00084
00083 L4_CV long
00090 l4re_ds_clear(const l4re_ds_t ds, l4_addr_t offset, unsigned long size)
00091 L4_NOTHROW;
00092
00098 L4_CV long
00099 l4re_ds_allocate(const l4re_ds_t ds,
00100 l4_addr_t offset, l4_size_t size) L4_NOTHROW;
00101
00107 L4_CV int
00108 l4re_ds_copy_in(const l4re_ds_t ds, l4_addr_t dst_offs, const
00109 l4re_ds_t src,
00110 l4_addr_t src_offs, unsigned long size) L4_NOTHROW;
00111
00116 L4_CV unsigned long
00117 l4re_ds_size(const l4re_ds_t ds) L4_NOTHROW;
00118
00123 L4_CV long
00124 l4re_ds_flags(const l4re_ds_t ds) L4_NOTHROW;
00125
00130 L4_CV int
00131 l4re_ds_info(const l4re_ds_t ds, l4re_ds_stats_t *stats)
00132 L4_NOTHROW;
00133
00151 L4_CV int
00152 l4re_ds_phys(const l4re_ds_t ds, l4_addr_t offset,
00153 l4_addr_t *phys_addr, l4_size_t *phys_size)
00154 L4_NOTHROW;
00155
00156 EXTERN_C_END

```

## 15.143 l4re/c/debug.h File Reference

Debug C interface.

```
#include <l4/sys/types.h>
Include dependency graph for debug.h:
```



## Functions

- long [l4re\\_debug\\_obj\\_debug](#) ([l4\\_cap\\_idx\\_t](#) srv, unsigned long function) [L4\\_NOTHROW](#)  
Call debug function of [L4Re](#) service.

### 15.143.1 Detailed Description

Debug C interface.

Definition in file [debug.h](#).

## 15.144 debug.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00029 #include <l4/sys/types.h>
```



## Functions

- long [l4re\\_dma\\_space\\_map](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_ds\\_t](#) src, [l4\\_addr\\_t](#) offset, [l4\\_size\\_t](#) \*size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) \*dma\_addr) [L4\\_NOTHROW](#)  
*Map the given part of this data space into the DMA address space.*
- long [l4re\\_dma\\_space\\_unmap](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) dma\_addr, [l4\\_size\\_t](#) size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir) [L4\\_NOTHROW](#)  
*Unmap the given part of this data space from the DMA address space.*
- long [l4re\\_dma\\_space\\_associate](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4\\_cap\\_idx\\_t](#) dma\_task, unsigned long attr) [L4\\_NOTHROW](#)  
*Associate a DMA task for a device to this Dma\_space.*
- long [l4re\\_dma\\_space\\_disassociate](#) ([l4re\\_dma\\_space\\_t](#) dma)  
*Disassociate the DMA task from this Dma\_space.*

### 15.145.1 Detailed Description

DMA space C interface.

Definition in file [dma\\_space.h](#).

### 15.145.2 Typedef Documentation

#### 15.145.2.1 l4re\_dma\_space\_dma\_addr\_t

```
typedef l4_uint64_t l4re_dma_space_dma_addr_t
```

Data type for DMA addresses.

Definition at line 62 of file [dma\\_space.h](#).

### 15.145.3 Enumeration Type Documentation

#### 15.145.3.1 l4re\_dma\_space\_direction

```
enum l4re_dma_space_direction
```

Direction of the DMA transfers.

#### Enumerator

|                              |                                       |
|------------------------------|---------------------------------------|
| L4RE_DMA_SPACE_BIDIRECTIONAL | device reads and writes to the memory |
| L4RE_DMA_SPACE_TO_DEVICE     | device reads the memory               |
| L4RE_DMA_SPACE_FROM_DEVICE   | device writes to the memory           |
| L4RE_DMA_SPACE_NONE          | device is coherently connected        |

Definition at line 37 of file [dma\\_space.h](#).

### 15.145.3.2 l4re\_dma\_space\_space\_attrbs

enum [l4re\\_dma\\_space\\_space\\_attrbs](#)

Attributes assigned to the DMA space when associated with a specific device.

See also

[Space\\_attrbs](#)

Enumerator

|                           |  |
|---------------------------|--|
| L4RE_DMA_SPACE_COHERENT   |  |
| L4RE_DMA_SPACE_PHYS_SPACE |  |

Definition at line 48 of file [dma\\_space.h](#).

## 15.146 dma\_space.h

```

00001
00005 /*
00006 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00029 #include <l4/sys/types.h>
00030 #include <l4/re/c/dataspace.h>
00031
00032 EXTERN_C_BEGIN
00033
00037 enum l4re_dma_space_direction
00038 {
00039 L4RE_DMA_SPACE_BIDIRECTIONAL,
00040 L4RE_DMA_SPACE_TO_DEVICE,
00041 L4RE_DMA_SPACE_FROM_DEVICE,
00042 L4RE_DMA_SPACE_NONE
00043 };
00044
00048 enum l4re_dma_space_space_attrbs
00049 {
00050 L4RE_DMA_SPACE_COHERENT = 1 << 0,
00051 L4RE_DMA_SPACE_PHYS_SPACE = 1 << 1,
00052 };
00053
00059 typedef l4_cap_idx_t l4re_dma_space_t;
00060
00062 typedef l4_uint64_t l4re_dma_space_dma_addr_t;
00063

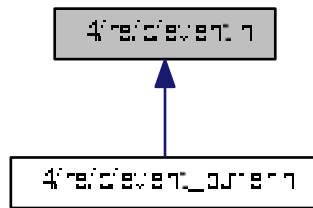
```

## 15.147 I4/re/c/event.h File Reference

```
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/event.h>
Include dependency graph for event.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_event\\_t](#)  
*Event structure used in buffer.*

## Functions

- long [l4re\\_event\\_get\\_buffer](#) (const [l4\\_cap\\_idx\\_t](#) server, const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get an event signal buffer.*
- long [l4re\\_event\\_get\\_num\\_streams](#) (const [l4\\_cap\\_idx\\_t](#) server) [L4\\_NOTHROW](#)  
*Get number of streams.*
- long [l4re\\_event\\_get\\_stream\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, int idx, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information on a stream.*
- long [l4re\\_event\\_get\\_stream\\_info\\_for\\_id](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) stream\_id, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get info for a stream given a stream id.*
- long [l4re\\_event\\_get\\_axis\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) id, unsigned naxes, unsigned const \*axis, [l4re\\_event\\_absinfo\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get Axis information for a stream.*

### 15.147.1 Detailed Description

Event C interface.

Definition in file [event.h](#).



## 15.148 event.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/re/c/dataspace.h>
00033 #include <l4/re/event.h>
00034
00035 EXTERN_C_BEGIN
00036
00040 typedef struct
00041 {
00042 long long time;
00043 unsigned short type;
00044 unsigned short code;
00045 int value;
00046 l4_umword_t stream_id;
00047 } l4re_event_t;
00048
00060 L4_CV long
00061 l4re_event_get_buffer(const l4_cap_idx_t server,
00062 const l4re_ds_t ds) L4_NOTHROW;
00063
00074 L4_CV long
00075 l4re_event_get_num_streams(const l4_cap_idx_t server)
00076 L4_NOTHROW;
00077
00089 L4_CV long
00090 l4re_event_get_stream_info(const l4_cap_idx_t server,
00091 int idx, l4re_event_stream_info_t *info) L4_NOTHROW;
00092
00105 L4_CV long
00106 l4re_event_get_stream_info_for_id(const
00107 l4_cap_idx_t server,
00108 l4_umword_t stream_id,
00109 l4re_event_stream_info_t *info) L4_NOTHROW;
00110
00125 L4_CV long
00126 l4re_event_get_axis_info(const l4_cap_idx_t server,
00127 l4_umword_t id,
00128 unsigned naxes, unsigned const *axis,
00129 l4re_event_absinfo_t *info) L4_NOTHROW;
00130
00130 EXTERN_C_END

```

## 15.149 l4/re/event.h File Reference

Events.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```



```

00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/sys/compiler.h>
00025 #include <l4/sys/l4int.h>
00026
00027 typedef struct L4_EXPORT_TYPE l4re_event_stream_id_t
00028 {
00029 l4_uint16_t bustype;
00030 l4_uint16_t vendor;
00031 l4_uint16_t product;
00032 l4_uint16_t version;
00033 } l4re_event_stream_id_t;
00034
00035 typedef struct L4_EXPORT_TYPE l4re_event_absinfo_t
00036 {
00037 l4_int32_t value;
00038 l4_int32_t min;
00039 l4_int32_t max;
00040 l4_int32_t fuzz;
00041 l4_int32_t flat;
00042 l4_int32_t resolution;
00043 } l4re_event_absinfo_t;
00044
00045 enum l4re_event_stream_max_values_t
00046 {
00047 L4RE_EVENT_EV_MAX = 0x1f,
00048 L4RE_EVENT_KEY_MAX = 0x1ff,
00049 L4RE_EVENT_REL_MAX = 0xf,
00050 L4RE_EVENT_ABS_MAX = 0x3f,
00051 L4RE_EVENT_PROP_MAX = 0x1f,
00052 L4RE_EVENT_SW_MAX = 0xf, // should be >= L4RE_SW_MAX
00053 };
00054
00055 enum l4re_event_stream_props_t
00056 {
00057 L4RE_EVENT_STREAM_CALIBRATE = 0x001,
00058 };
00059
00060
00061 #define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)
00062
00063 typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
00064 {
00065 l4_umword_t stream_id;
00066 char name[32];
00067 char phys[32];
00068 l4re_event_stream_id_t id;
00069
00070 unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];
00071
00072 unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];
00073 unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00074 unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
00075 unsigned long absbits[__UNUM_B(L4RE_EVENT_ABS_MAX)];
00076 unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00077
00078 } l4re_event_stream_info_t;
00079
00080 typedef struct L4_EXPORT_TYPE l4re_event_stream_state_t
00081 {
00082 unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00083 unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00084 } l4re_event_stream_state_t;
00085
00086 #undef __UNUM_B
00087

```

## 15.151 l4/re/c/log.h File Reference

Log C interface.



```

00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/re/env.h>
00025
00026 EXTERN_C_BEGIN
00027
00028 L4_CV L4_INLINE void
00029 l4re_log_print(char const *string) L4_NOTHROW;
00030
00031 L4_CV L4_INLINE void
00032 l4re_log_printn(char const *string, int len) L4_NOTHROW;
00033
00034
00035 L4_CV void
00036 l4re_log_print_srv(const l4_cap_idx_t logcap,
00037 char const *string) L4_NOTHROW;
00038
00039 L4_CV void
00040 l4re_log_printn_srv(const l4_cap_idx_t logcap,
00041 char const *string, int len) L4_NOTHROW;
00042
00043 /***** Implementations *****/
00044
00045 L4_CV L4_INLINE void
00046 l4re_log_print(char const *string) L4_NOTHROW
00047 {
00048 l4re_log_print_srv(l4re_global_env->log, string);
00049 }
00050
00051 L4_CV L4_INLINE void
00052 l4re_log_printn(char const *string, int len) L4_NOTHROW
00053 {
00054 l4re_log_printn_srv(l4re_global_env->log, string, len);
00055 }
00056
00057 EXTERN_C_END

```

## 15.153 l4/re/c/mem\_alloc.h File Reference

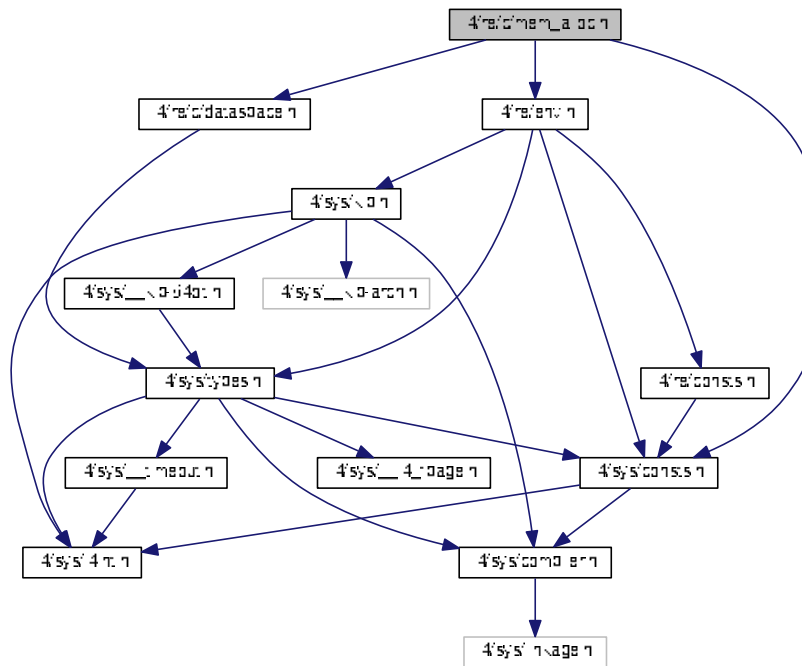
Memory allocator C interface.

```

#include <l4/re/env.h>
#include <l4/sys/consts.h>
#include <l4/re/c/dataspace.h>

```

Include dependency graph for `mem_alloc.h`:



## Enumerations

- enum [l4re\\_ma\\_flags](#)  
*Flags for requesting memory at the memory allocator.*

## Functions

- long [l4re\\_ma\\_alloc](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_free](#) ([l4re\\_ds\\_t](#) const mem) [L4\\_NOTHROW](#)  
*Free memory.*
- long [l4re\\_ma\\_alloc\\_align\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_free\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, [l4re\\_ds\\_t](#) const mem) [L4\\_NOTHROW](#)  
*Free memory.*

### 15.153.1 Detailed Description

Memory allocator C interface.

Definition in file [mem\\_alloc.h](#).

## 15.154 mem\_alloc.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/re/env.h>
00025 #include <l4/sys/consts.h>
00026
00027 #include <l4/re/c/dataspace.h>
00028
00035 EXTERN_C_BEGIN
00036
00042 enum l4re_ma_flags {
00043 L4RE_MA_CONTINUOUS = 0x01,
00044 L4RE_MA_PINNED = 0x02,
00045 L4RE_MA_SUPER_PAGES = 0x04,
00046 };
00047
00048
00066 L4_CV L4_INLINE long
00067 l4re_ma_alloc(long size, l4re_ds_t const mem,
00068 unsigned long flags) L4_NOTHROW;
00069
00091 L4_CV L4_INLINE long
00092 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00093 unsigned long flags, unsigned long align) L4_NOTHROW;
00094
00106 /* Deprecation message added Q4 2016.
00107 * Remove together with L4Re::Mem_alloc::free() */
00108 L4_CV L4_INLINE long
00109 l4re_ma_free(l4re_ds_t const mem) L4_NOTHROW
00110 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See
00111 documentation for replacement options.");
00112
00113
00114
00133 L4_CV long
00134 l4re_ma_alloc_align_srv(l4_cap_idx_t srv, long size,
00135 l4re_ds_t const mem, unsigned long flags,
00136 unsigned long align) L4_NOTHROW;
00137
00150 /* Deprecation message added Q4 2016.
00151 * Remove together with L4Re::Mem_alloc::free() */
00152 L4_CV long
00153 l4re_ma_free_srv(l4_cap_idx_t srv, l4re_ds_t const mem)
00154 L4_NOTHROW
00155 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See
00156 documentation for replacement options.");
00157
00158
00159
00160 /***** Implementation *****/
00161
00162 /* Just warn actual users, but not for internal implementations */
00163 #pragma GCC diagnostic push
00164 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00165
00166 L4_CV L4_INLINE long
00167 l4re_ma_alloc(long size, l4re_ds_t const mem,
00168 unsigned long flags) L4_NOTHROW
00169 {
00170 return l4re_ma_alloc_align_srv(l4re_global_env->
00171 mem_alloc, size, mem,
00172 flags, 0);
00173 }
00174

```





## Functions

- long [l4re\\_ns\\_query\\_to\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap, int timeout) [L4\\_NOTHROW](#)  
*Query the name space for the object named by name.*
- long [l4re\\_ns\\_query\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap) [L4\\_NOTHROW](#)  
*Query the name space for the object named by name.*
- long [l4re\\_ns\\_register\\_obj\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const obj, unsigned flags) [L4\\_NOTHROW](#)  
*Register an object with a name.*

### 15.155.1 Detailed Description

Namespace functions, C interface.

Definition in file [namespace.h](#).

## 15.156 namespace.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00031 #include <l4/re/env.h>
00032
00039 enum l4re_ns_register_flags {
00040 L4RE_NS_REGISTER_RO = L4_FPAGE_RO,
00041 L4RE_NS_REGISTER_DIR = 0x10,
00042 L4RE_NS_REGISTER_RW = L4_FPAGE_RX,
00043 L4RE_NS_REGISTER_RWS = L4_FPAGE_RWX,
00044 L4RE_NS_REGISTER_S = L4_FPAGE_W,
00045 };
00046
00047 EXTERN_C_BEGIN
00048
00053 typedef l4_cap_idx_t l4re_namespace_t;
00054
00055
00056
00065 L4_CV long
00066 l4re_ns_query_to_srv(l4re_namespace_t srv, char const *name,
00067 l4_cap_idx_t const cap, int timeout) L4_NOTHROW;
00068
00085 L4_CV L4_INLINE long
00086 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00087 l4_cap_idx_t const cap) L4_NOTHROW;
00088
00096 L4_CV long
00097 l4re_ns_register_obj_srv(l4re_namespace_t srv, char const *name,
00098 l4_cap_idx_t const obj, unsigned flags)
00099 L4_NOTHROW;
00099
00100

```



## Functions

- `int l4re_rm_reserve_area (l4_addr_t *start, unsigned long size, unsigned flags, unsigned char align) L4_NOTHROW`
- `int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW`
- `int l4re_rm_attach (void **start, unsigned long size, unsigned long flags, l4re_ds_t const mem, l4_addr_t offs, unsigned char align) L4_NOTHROW`
- `int l4re_rm_detach (void *addr) L4_NOTHROW`  
*Detach and unmap in current task.*
- `int l4re_rm_detach_ds (void *addr, l4re_ds_t *ds) L4_NOTHROW`  
*Detach, unmap and return affected dataspace in current task.*
- `int l4re_rm_detach_unmap (l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW`  
*Detach and unmap in specified task.*
- `int l4re_rm_detach_ds_unmap (void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`  
*Detach and unmap in specified task.*
- `int l4re_rm_find (l4_addr_t *addr, unsigned long *size, l4_addr_t *offset, unsigned *flags, l4re_ds_t *m) L4_NOTHROW`
- `void l4re_rm_show_lists (void) L4_NOTHROW`  
*Dump region map internal data structures.*
- `int l4re_rm_reserve_area_srv (l4_cap_idx_t rm, l4_addr_t *start, unsigned long size, unsigned flags, unsigned char align) L4_NOTHROW`
- `int l4re_rm_free_area_srv (l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW`
- `int l4re_rm_attach_srv (l4_cap_idx_t rm, void **start, unsigned long size, unsigned long flags, l4re_ds_t const mem, l4_addr_t offs, unsigned char align) L4_NOTHROW`
- `int l4re_rm_detach_srv (l4_cap_idx_t rm, l4_addr_t addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`
- `int l4re_rm_find_srv (l4_cap_idx_t rm, l4_addr_t *addr, unsigned long *size, l4_addr_t *offset, unsigned *flags, l4re_ds_t *m) L4_NOTHROW`
- `void l4re_rm_show_lists_srv (l4_cap_idx_t rm) L4_NOTHROW`  
*Dump region map internal data structures.*

### 15.157.1 Detailed Description

Region map interface, C interface.

Definition in file [rm.h](#).

## 15.158 rm.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once

```

```

00024
00031 #include <l4/re/env.h>
00032 #include <l4/re/c/dataspace.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 enum l4re_rm_flags_t {
00041 L4RE_RM_READ_ONLY = 0x01,
00042 L4RE_RM_NO_ALIAS = 0x02,
00043 L4RE_RM_PAGER = 0x04,
00044 L4RE_RM_RESERVED = 0x08,
00046 L4RE_RM_CACHING_SHIFT = 8,
00049 L4RE_RM_CACHING_DS_SHIFT = L4RE_RM_CACHING_SHIFT -
 L4RE_DS_MAP_CACHING_SHIFT,
00050
00052 L4RE_RM_CACHING = L4RE_DS_MAP_CACHING_MASK <<
 L4RE_RM_CACHING_DS_SHIFT,
00053
00054 L4RE_RM_REGION_FLAGS = L4RE_RM_CACHING | 0x0f,
00057 L4RE_RM_CACHE_NORMAL = L4RE_DS_MAP_NORMAL <<
 L4RE_RM_CACHING_DS_SHIFT,
00058
00060 L4RE_RM_CACHE_BUFFERED = L4RE_DS_MAP_BUFFERABLE <<
 L4RE_RM_CACHING_DS_SHIFT,
00061
00063 L4RE_RM_CACHE_UNCACHED = L4RE_DS_MAP_UNCACHEABLE <<
 L4RE_RM_CACHING_DS_SHIFT,
00064
00065 L4RE_RM_OVERMAP = 0x10,
00066 L4RE_RM_SEARCH_ADDR = 0x20,
00067 L4RE_RM_IN_AREA = 0x40,
00068 L4RE_RM_EAGER_MAP = 0x80,
00069 L4RE_RM_ATTACH_FLAGS = 0xf0,
00070 };
00071
00072
00073
00081 L4_CV L4_INLINE int
00082 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00083 unsigned flags, unsigned char align) L4_NOTHROW;
00084
00092 L4_CV L4_INLINE int
00093 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;
00094
00103 L4_CV L4_INLINE int
00104 l4re_rm_attach(void **start, unsigned long size, unsigned long flags,
00105 l4re_ds_t const mem, l4_addr_t offs,
00106 unsigned char align) L4_NOTHROW;
00107
00108
00119 L4_CV L4_INLINE int
00120 l4re_rm_detach(void *addr) L4_NOTHROW;
00121
00134 L4_CV L4_INLINE int
00135 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds)
 L4_NOTHROW;
00136
00148 L4_CV L4_INLINE int
00149 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task)
 L4_NOTHROW;
00150
00163 L4_CV L4_INLINE int
00164 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00165 l4_cap_idx_t task) L4_NOTHROW;
00166
00167
00173 L4_CV L4_INLINE int
00174 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
 l4_addr_t *offset,
00175 unsigned *flags, l4re_ds_t *m) L4_NOTHROW;
00176
00183 L4_CV L4_INLINE void
00184 l4re_rm_show_lists(void) L4_NOTHROW;
00185
00186
00187 /*
00188 * Variants of functions that also take a capability of the region map
00189 * service.
00190 */
00191
00192
00197 L4_CV int
00198 l4re_rm_reserve_area_srv(l4_cap_idx_t rm,
 l4_addr_t *start, unsigned long size,
00199 unsigned flags, unsigned char align) L4_NOTHROW;
00200
00205 L4_CV int

```

```

00206 l4re_rm_free_area_srv(l4_cap_idx_t rm,
00207 l4_addr_t addr) L4_NOTHROW;
00207
00212 L4_CV int
00213 l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00214 unsigned long flags, l4re_ds_t const mem, l4_addr_t offs,
00215 unsigned char align) L4_NOTHROW;
00216
00217
00222 L4_CV int
00223 l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
00224 l4re_ds_t *ds, l4_cap_idx_t task)
00225 L4_NOTHROW;
00225
00226
00231 L4_CV int
00232 l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
00233 unsigned long *size, l4_addr_t *offset,
00234 unsigned *flags, l4re_ds_t *m) L4_NOTHROW;
00235
00240 L4_CV void
00241 l4re_rm_show_lists_srv(l4_cap_idx_t rm)
00242 L4_NOTHROW;
00242
00243
00244 /***** Implementations *****/
00245
00246 L4_CV L4_INLINE int
00247 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00248 unsigned flags, unsigned char align) L4_NOTHROW
00249 {
00250 return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
00251 flags, align);
00252 }
00253
00254 L4_CV L4_INLINE int
00255 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
00256 {
00257 return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
00258 }
00259
00260 L4_CV L4_INLINE int
00261 l4re_rm_attach(void **start, unsigned long size, unsigned long flags,
00262 l4re_ds_t const mem, l4_addr_t offs,
00263 unsigned char align) L4_NOTHROW
00264 {
00265 return l4re_rm_attach_srv(l4re_global_env->rm, start, size,
00266 flags, mem, offs, align);
00267 }
00268
00269
00270 L4_CV L4_INLINE int
00271 l4re_rm_detach(void *addr) L4_NOTHROW
00272 {
00273 return l4re_rm_detach_srv(l4re_global_env->rm,
00274 (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
00275 }
00276
00277 L4_CV L4_INLINE int
00278 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task)
00279 L4_NOTHROW
00280 {
00281 return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
00282 }
00282
00283 L4_CV L4_INLINE int
00284 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds)
00285 L4_NOTHROW
00286 {
00287 return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00288 ds, L4_BASE_TASK_CAP);
00289 }
00289
00290 L4_CV L4_INLINE int
00291 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00292 l4_cap_idx_t task) L4_NOTHROW
00293 {
00294 return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00295 ds, task);
00296 }
00296
00297 L4_CV L4_INLINE int
00298 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00299 l4_addr_t *offset,
00300 unsigned *flags, l4re_ds_t *m) L4_NOTHROW
00301 {
00302 return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);

```

```

00302 }
00303
00304 L4_CV L4_INLINE void
00305 l4re_rm_show_lists(void) L4_NOTHROW
00306 {
00307 l4re_rm_show_lists_srv(l4re_global_env->rm);
00308 }
00309
00310 EXTERN_C_END

```

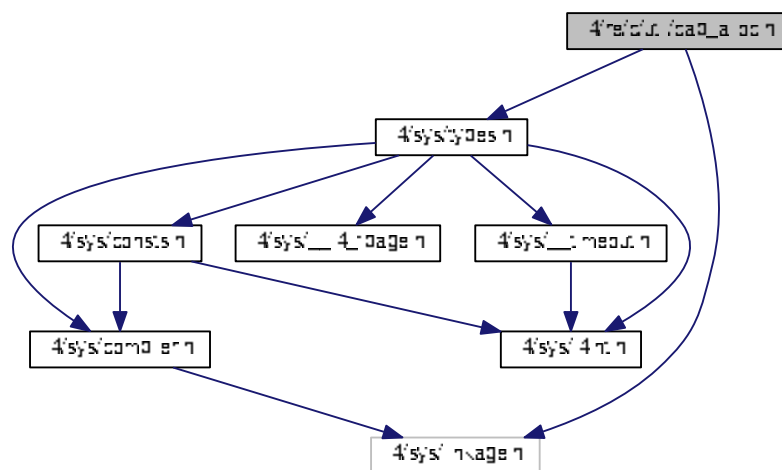
## 15.159 l4re/c/util/cap\_alloc.h File Reference

Capability allocator C interface.

```

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
Include dependency graph for cap_alloc.h:

```



## Functions

- [l4\\_cap\\_idx\\_t l4re\\_util\\_cap\\_alloc](#) (void) [L4\\_NOTHROW](#)  
Get free capability index at capability allocator.
- void [l4re\\_util\\_cap\\_free](#) ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
Return capability index to capability allocator.
- void [l4re\\_util\\_cap\\_free\\_um](#) ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
Return capability index to capability allocator, and unmaps the object.
- long [l4re\\_util\\_cap\\_last](#) (void) [L4\\_NOTHROW](#)  
Return last capability index the allocator can return.

### 15.159.1 Detailed Description

Capability allocator C interface.

Definition in file [cap\\_alloc.h](#).

## 15.160 cap\_alloc.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/sys/linkage.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 L4_CV l4_cap_idx_t
00041 l4re_util_cap_alloc(void) L4_NOTHROW;
00042
00047 L4_CV void
00048 l4re_util_cap_free(l4_cap_idx_t cap) L4_NOTHROW;
00049
00055 L4_CV void
00056 l4re_util_cap_free_um(l4_cap_idx_t cap)
 L4_NOTHROW;
00057
00063 L4_CV long
00064 l4re_util_cap_last(void) L4_NOTHROW;
00065
00066 EXTERN_C_END

```

## 15.161 l4re/c/util/kumem\_alloc.h File Reference

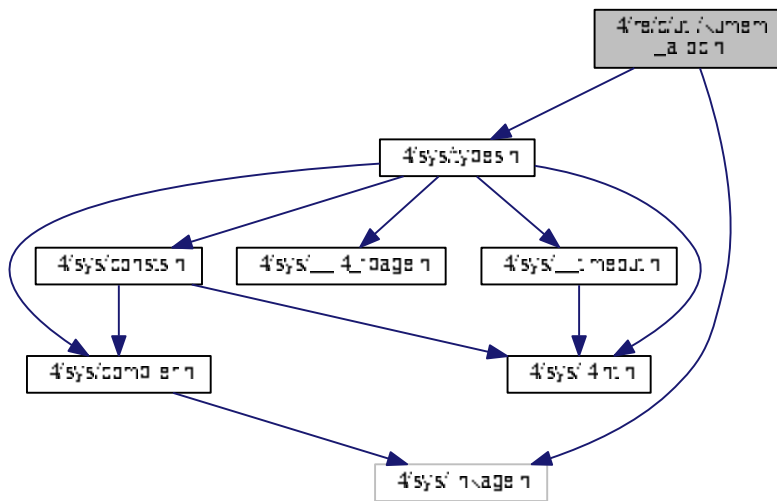
Kumem allocator utility C interface.

```

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

```

Include dependency graph for `kumem_alloc.h`:



## Functions

- `int l4re_util_kumem_alloc (l4_addr_t *mem, unsigned pages_order, l4_cap_idx_t task, l4_cap_idx_t rm) L4↔_NOTHROW`  
Allocate state area.

### 15.161.1 Detailed Description

Kumem allocator utility C interface.

Definition in file `kumem_alloc.h`.

### 15.161.2 Function Documentation

#### 15.161.2.1 l4re\_util\_kumem\_alloc()

```
int l4re_util_kumem_alloc (
 l4_addr_t * mem,
 unsigned pages_order,
 l4_cap_idx_t task,
 l4_cap_idx_t rm)
```

Allocate state area.



## Parameters

|     |                    |                                            |
|-----|--------------------|--------------------------------------------|
| out | <i>mem</i>         | Pointer to memory that has been allocated. |
|     | <i>pages_order</i> | Size to allocate, in log2 pages.           |
|     | <i>task</i>        | Task to use for allocation.                |
|     | <i>rm</i>          | Region manager to use for allocation.      |

## Return values

|    |                       |
|----|-----------------------|
| 0  | for success           |
| <0 | error code on failure |

## Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

## Examples:

[examples/sys/aliens/main.c](#).

## 15.162 kumem\_alloc.h

```

00001
00005 /*
00006 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/sys/linkage.h>
00033
00034 EXTERN_C_BEGIN
00035
00039 L4_CV int
00040 l4re_util_kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00041 l4_cap_idx_t task, l4_cap_idx_t rm)
00042 L4_NOTHROW;
00043 EXTERN_C_END

```

## 15.163 l4re/c/util/video/goos\_fb.h File Reference

Framebuffer utility functionality.



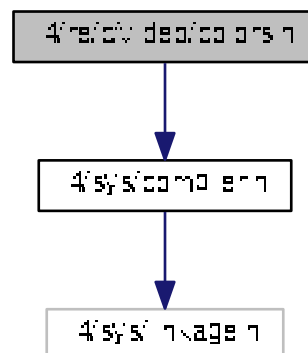
```

00022 #pragma once
00023
00024 #include <l4/sys/compiler.h>
00025 #include <l4/re/c/video/goos.h>
00026 #include <l4/sys/err.h>
00027 #include <l4/re/c/dataspace.h>
00028
00029 EXTERN_C_BEGIN
00030
00031 typedef struct
00032 {
00033 unsigned long _obj_buf[6];
00034 } l4re_util_video_goos_fb_t;
00035
00036 L4_CV int
00037 l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
00038 char const *name) L4_NOTHROW;
00039
00040 L4_CV void
00041 l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00042
00043 L4_CV int
00044 l4re_util_video_goos_fb_view_info(l4re_util_video_goos_fb_t *goosfb,
00045 l4re_video_view_info_t *info)
00046 L4_NOTHROW;
00047
00048 L4_CV void *
00049 l4re_util_video_goos_fb_attach_buffer(l4re_util_video_goos_fb_t *goosfb)
00050 L4_NOTHROW;
00051
00052 L4_CV int
00053 l4re_util_video_goos_fb_refresh(l4re_util_video_goos_fb_t *goosfb,
00054 int x, int y, int w, int h) L4_NOTHROW;
00055
00056 L4_CV l4re_ds_t
00057 l4re_util_video_goos_fb_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00058
00059 L4_CV l4_cap_idx_t
00060 l4re_util_video_goos_fb_goos(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00061
00062 EXTERN_C_END

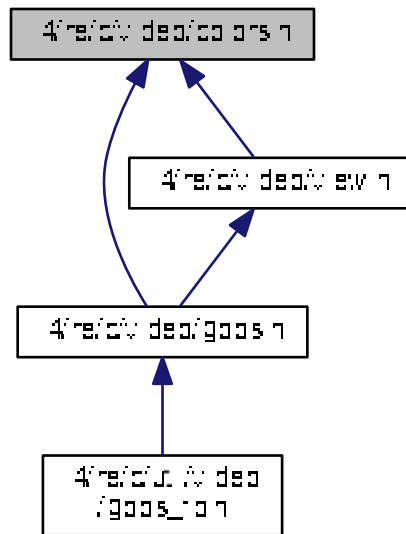
```

## 15.165 l4/re/c/video/colors.h File Reference

#include <l4/sys/compiler.h>  
 Include dependency graph for colors.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- struct [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*

## Typedefs

- typedef struct [l4re\\_video\\_color\\_component\\_t](#) [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- typedef struct [l4re\\_video\\_pixel\\_info\\_t](#) [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*

### 15.165.1 Detailed Description

#### Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ interface where possible.

Definition in file [colors.h](#).

## 15.166 colors.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00031 typedef struct l4re_video_color_component_t
00032 {
00033 unsigned char size;
00034 unsigned char shift;
00035 } __attribute__((packed)) l4re_video_color_component_t;
00036
00041 typedef struct l4re_video_pixel_info_t
00042 {
00043 l4re_video_color_component_t r, g, b, a;
00044 unsigned char bytes_per_pixel;
00045 } l4re_video_pixel_info_t;
00046
00047 EXTERN_C_BEGIN
00048
00049 L4_INLINE L4_CV int
00050 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p)
00051 L4_NOTHROW;
00052
00053 /* *****
00054 * Implementations */
00055
00056 L4_INLINE L4_CV int
00057 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW
00058 {
00059 return p->r.size + p->b.size + p->g.size + p->a.size;
00060 }
00061 EXTERN_C_END

```

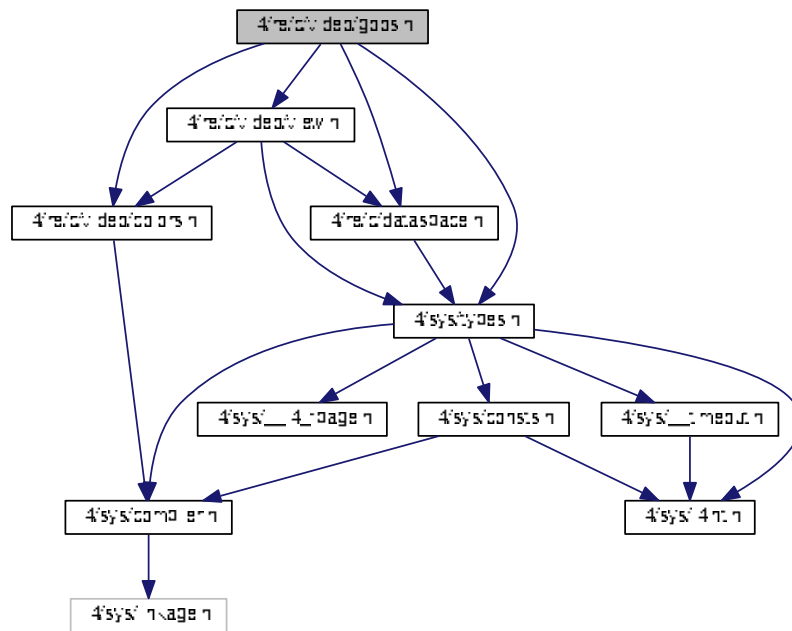
## 15.167 l4/re/c/video/goos.h File Reference

```

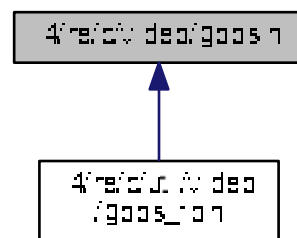
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
#include <l4/re/c/video/view.h>

```

Include dependency graph for goos.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4re_video_goos_info_t`  
*Goos information structure.*

## Typedefs

- `typedef l4_cap_idx_t l4re_video_goos_t`  
*Goos object type.*

## Enumerations

- enum `l4re_video_goos_info_flags_t` { `F_l4re_video_goos_auto_refresh` = 0x01, `F_l4re_video_goos_pointer` = 0x02, `F_l4re_video_goos_dynamic_views` = 0x04, `F_l4re_video_goos_dynamic_buffers` = 0x08 }

*Flags of information on the goos.*

## Functions

- int `l4re_video_goos_info` (`l4re_video_goos_t` goos, `l4re_video_goos_info_t` \*ginfo) `L4_NOTHROW`  
*Get information on a goos.*
- int `l4re_video_goos_refresh` (`l4re_video_goos_t` goos, int x, int y, int w, int h) `L4_NOTHROW`  
*Flush a rectangle of pixels of the goos screen.*
- int `l4re_video_goos_create_buffer` (`l4re_video_goos_t` goos, unsigned long size, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Create a new buffer (memory buffer) for pixel data.*
- int `l4re_video_goos_delete_buffer` (`l4re_video_goos_t` goos, unsigned idx) `L4_NOTHROW`  
*Delete a pixel buffer.*
- int `l4re_video_goos_get_static_buffer` (`l4re_video_goos_t` goos, unsigned idx, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Get the data-space capability of the static pixel buffer.*
- int `l4re_video_goos_create_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Create a new view (.*
- int `l4re_video_goos_delete_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Delete a view.*
- int `l4re_video_goos_get_view` (`l4re_video_goos_t` goos, unsigned idx, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Get a view for the given index.*

### 15.167.1 Detailed Description

#### Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [goos.h](#).

## 15.168 goos.h

```
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
```

```

00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/re/c/dataspace.h>
00027 #include <l4/re/c/video/colors.h>
00028 #include <l4/re/c/video/view.h>
00029
00039 enum l4re_video_goos_info_flags_t
00040 {
00041 F_l4re_video_goos_auto_refresh = 0x01,
00042 F_l4re_video_goos_pointer = 0x02,
00043 F_l4re_video_goos_dynamic_views = 0x04,
00044 F_l4re_video_goos_dynamic_buffers = 0x08,
00045 };
00046
00051 typedef struct
00052 {
00053 unsigned long width;
00054 unsigned long height;
00055 unsigned flags;
00056 unsigned num_static_views;
00057 unsigned num_static_buffers;
00058 l4re_video_pixel_info_t pixel_info;
00059 } l4re_video_goos_info_t;
00060
00065 typedef l4_cap_idx_t l4re_video_goos_t;
00066
00067 EXTERN_C_BEGIN
00068
00080 L4_CV int
00081 l4re_video_goos_info(l4re_video_goos_t goos,
00082 l4re_video_goos_info_t *ginfo)
00083 L4_NOTHROW;
00084
00093 L4_CV int
00094 l4re_video_goos_refresh(l4re_video_goos_t goos, int x, int y, int w
00095 ,
00096 int h) L4_NOTHROW;
00097
00108 L4_CV int
00109 l4re_video_goos_create_buffer(l4re_video_goos_t goos,
00110 unsigned long size,
00111 l4_cap_idx_t buffer) L4_NOTHROW;
00112
00119 L4_CV int
00120 l4re_video_goos_delete_buffer(l4re_video_goos_t goos,
00121 unsigned idx) L4_NOTHROW;
00122
00132 L4_CV int
00133 l4re_video_goos_get_static_buffer(
00134 l4re_video_goos_t goos, unsigned idx,
00135 l4_cap_idx_t buffer) L4_NOTHROW;
00136
00142 L4_CV int
00143 l4re_video_goos_create_view(l4re_video_goos_t goos,
00144 l4re_video_view_t *view) L4_NOTHROW;
00145
00153 L4_CV int
00154 l4re_video_goos_delete_view(l4re_video_goos_t goos,
00155 l4re_video_view_t *view) L4_NOTHROW;
00156
00157
00170 L4_CV int
00171 l4re_video_goos_get_view(l4re_video_goos_t goos, unsigned idx,
00172 l4re_video_view_t *view) L4_NOTHROW;
00173
00174 EXTERN_C_END

```

## 15.169 l4/re/c/video/view.h File Reference

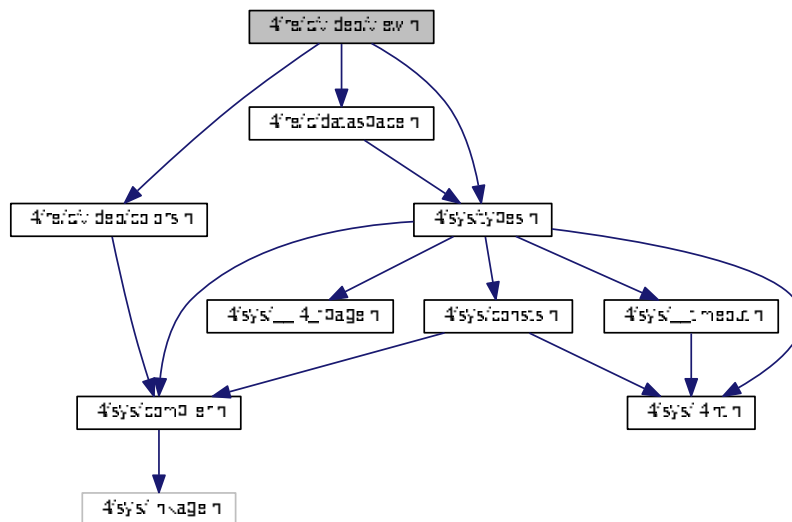
```

#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>

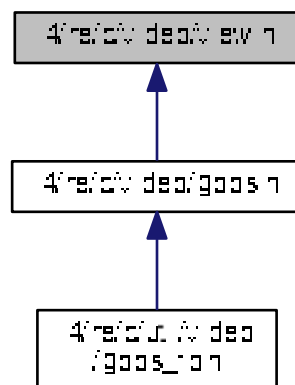
```



Include dependency graph for view.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4re_video_view_info_t`  
*View information structure.*
- struct `l4re_video_view_t`  
*C representation of a goos view.*

## Typedefs

- typedef struct [l4re\\_video\\_view\\_info\\_t](#) [l4re\\_video\\_view\\_info\\_t](#)

*View information structure.*

- typedef struct [l4re\\_video\\_view\\_t](#) [l4re\\_video\\_view\\_t](#)

*C representation of a goos view.*

## Enumerations

- enum [l4re\\_video\\_view\\_info\\_flags\\_t](#) {  
[F\\_l4re\\_video\\_view\\_none](#) = 0x00, [F\\_l4re\\_video\\_view\\_set\\_buffer](#) = 0x01, [F\\_l4re\\_video\\_view\\_set\\_buffer\\_↔](#)  
[offset](#) = 0x02, [F\\_l4re\\_video\\_view\\_set\\_bytes\\_per\\_line](#) = 0x04,  
[F\\_l4re\\_video\\_view\\_set\\_pixel](#) = 0x08, [F\\_l4re\\_video\\_view\\_set\\_position](#) = 0x10, [F\\_l4re\\_video\\_view\\_dyn\\_↔](#)  
[allocated](#) = 0x20, [F\\_l4re\\_video\\_view\\_set\\_background](#) = 0x40,  
[F\\_l4re\\_video\\_view\\_set\\_flags](#) = 0x80, [F\\_l4re\\_video\\_view\\_above](#) = 0x01000, [F\\_l4re\\_video\\_view\\_flags\\_mask](#)  
= 0xff000 }

*Flags of information on a view.*

## Functions

- int [l4re\\_video\\_view\\_refresh](#) ([l4re\\_video\\_view\\_t](#) \*view, int x, int y, int w, int h) [L4\\_NOTHROW](#)  
*Flush the given rectangle of pixels of the given view.*
- int [l4re\\_video\\_view\\_get\\_info](#) ([l4re\\_video\\_view\\_t](#) \*view, [l4re\\_video\\_view\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Retrieve information about the given view.*
- int [l4re\\_video\\_view\\_set\\_info](#) ([l4re\\_video\\_view\\_t](#) \*view, [l4re\\_video\\_view\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Set properties of the view.*
- int [l4re\\_video\\_view\\_set\\_viewport](#) ([l4re\\_video\\_view\\_t](#) \*view, int x, int y, int w, int h, unsigned long bofs) [L4\\_↔](#)  
[NOTHROW](#)  
*Set the viewport parameters of a view.*
- int [l4re\\_video\\_view\\_stack](#) ([l4re\\_video\\_view\\_t](#) \*view, [l4re\\_video\\_view\\_t](#) \*pivot, int behind) [L4\\_NOTHROW](#)  
*Change the stacking order in the stack of visible views.*

### 15.169.1 Detailed Description

#### Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [view.h](#).

## 15.170 view.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/re/c/dataspace.h>
00027 #include <l4/re/c/video/colors.h>
00028
00033 enum l4re_video_view_info_flags_t
00034 {
00035 F_l4re_video_view_none = 0x00,
00036 F_l4re_video_view_set_buffer = 0x01,
00037 F_l4re_video_view_set_buffer_offset = 0x02,
00038 F_l4re_video_view_set_bytes_per_line = 0x04,
00039 F_l4re_video_view_set_pixel = 0x08,
00040 F_l4re_video_view_set_position = 0x10,
00041 F_l4re_video_view_dyn_allocated = 0x20,
00042 F_l4re_video_view_set_background = 0x40,
00043 F_l4re_video_view_set_flags = 0x80,
00044 F_l4re_video_view_fully_dynamic = F_l4re_video_view_set_buffer
00045 | F_l4re_video_view_set_buffer_offset
00046 | F_l4re_video_view_set_bytes_per_line
00047 | F_l4re_video_view_set_pixel
00048 | F_l4re_video_view_set_position
00049 | F_l4re_video_view_dyn_allocated,
00050
00051 F_l4re_video_view_above = 0x01000,
00052 F_l4re_video_view_flags_mask = 0xff000,
00053 };
00054
00059 typedef struct l4re_video_view_info_t
00060 {
00061 unsigned flags;
00062 unsigned view_index;
00063 unsigned long xpos, ypos, width, height;
00064 unsigned long buffer_offset;
00065 unsigned long bytes_per_line;
00066 l4re_video_pixel_info_t pixel_info;
00067 unsigned buffer_index;
00068 } l4re_video_view_info_t;
00069
00070
00078 typedef struct l4re_video_view_t
00079 {
00080 l4_cap_idx_t goos;
00081 unsigned idx;
00082 } l4re_video_view_t;
00083
00084
00085 EXTERN_C_BEGIN
00086
00096 L4_CV int
00097 l4re_video_view_refresh(l4re_video_view_t *view, int x, int y, int
00098 w,
00099 int h) L4_NOTHROW;
00100
00106 L4_CV int
00107 l4re_video_view_get_info(l4re_video_view_t *view,
00108 l4re_video_view_info_t *info)
00109 L4_NOTHROW;
00110
00120 L4_CV int
00121 l4re_video_view_set_info(l4re_video_view_t *view,
00122 l4re_video_view_info_t *info)
00123 L4_NOTHROW;
00139 L4_CV int
00140 l4re_video_view_set_viewport(l4re_video_view_t *view, int x,

```

```

 int y, int w,
00141 int h, unsigned long bofs) L4_NOTHROW;
00142
00152 L4_CV int
00153 l4re_video_view_stack(l4re_video_view_t *view,
 l4re_video_view_t *pivot,
00154 int behind) L4_NOTHROW;
00155
00156 EXTERN_C_END
00157

```

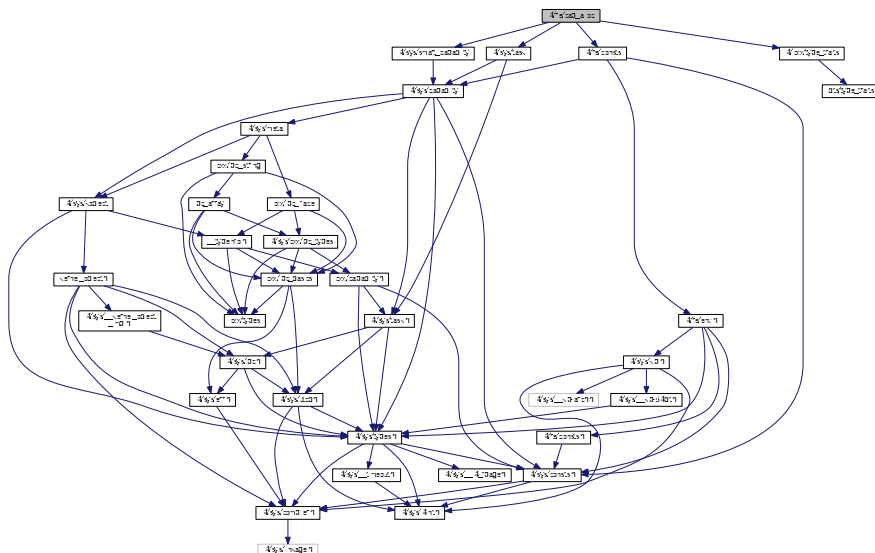
## 15.171 l4/re/cap\_alloc File Reference

Abstract capability-allocator interface.

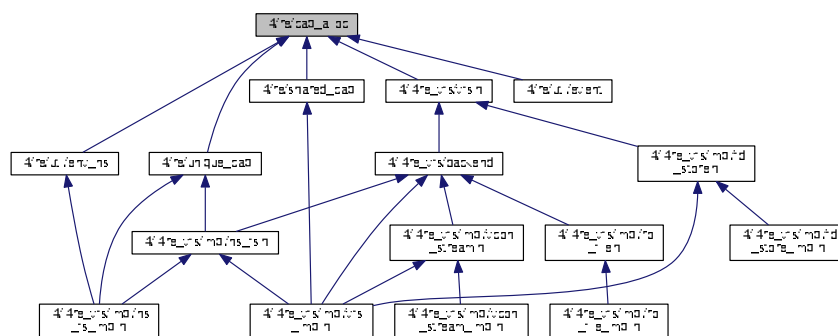
```

#include <l4/sys/task>
#include <l4/sys/smart_capability>
#include <l4/re/consts>
#include <l4/cxx/type_traits>
Include dependency graph for cap_alloc:

```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Cap\\_alloc](#)  
*Capability allocator interface.*
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for Auto\_cap and Auto\_del\_cap.*
- class [L4Re::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for Ref\_cap and Ref\_del\_cap.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.171.1 Detailed Description

Abstract capability-allocator interface.

Definition in file [cap\\_alloc](#).

## 15.172 cap\_alloc

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028 #include <l4/sys/smart_capability>
00029 #include <l4/re/consts>
00030 #include <l4/cxx/type_traits>
00031
00032 namespace L4Re {
00033
00041 class Cap_alloc
00042 {
00043 private:
00044 void operator = (Cap_alloc const &);
00045
00046 protected:
00047 Cap_alloc(Cap_alloc const &) {}
00048 Cap_alloc() {}
00049
00050 public:
00051
00056 virtual L4::Cap<void> alloc() throw() = 0;
00057 virtual void take(L4::Cap<void> cap) throw() = 0;
00058
00063 template< typename T >

```

```

00064 L4::Cap<T> alloc() throw()
00065 { return L4::cap_cast<T>(alloc()); }
00066
00073 virtual void free(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00074 unsigned unmap_flags = L4_FP_ALL_SPACES) throw() = 0;
00075 virtual bool release(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00076 unsigned unmap_flags = L4_FP_ALL_SPACES) throw() = 0;
00077
00081 virtual ~Cap_alloc() = 0;
00082
00088 template< typename CAP_ALLOC >
00089 static inline L4Re::Cap_alloc *
00090 get_cap_alloc(CAP_ALLOC &ca)
00091 {
00092 struct CA : public L4Re::Cap_alloc
00093 {
00094 CAP_ALLOC &_ca;
00095 L4::Cap<void> alloc() throw() override { return _ca.alloc(); }
00096 void take(L4::Cap<void> cap) throw() override { _ca.take(cap); }
00097
00098 void free(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00099 unsigned unmap_flags = L4_FP_ALL_SPACES) throw() override
00100 { _ca.free(cap, task, unmap_flags); }
00101
00102 bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00103 unsigned unmap_flags) throw() override
00104 { return _ca.release(cap, task, unmap_flags); }
00105
00106 void operator delete(void *) {}
00107
00108 CA(CAP_ALLOC &ca) : _ca(ca) {}
00109 };
00110
00111 static CA _ca(ca);
00112 return &_ca;
00113 }
00114 };
00115
00116 template<typename ALLOC>
00117 struct Cap_alloc_t : ALLOC, L4Re::Cap_alloc
00118 {
00119 template<typename ...ARGS>
00120 Cap_alloc_t(ARGS &&...args) : ALLOC(cxx::forward<ARGS>(args)...) {}
00121
00122 L4::Cap<void> alloc() throw() override { return ALLOC::alloc(); }
00123 void take(L4::Cap<void> cap) throw() override { ALLOC::take(cap); }
00124
00125 void free(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00126 unsigned unmap_flags = L4_FP_ALL_SPACES) throw() override
00127 { ALLOC::free(cap, task, unmap_flags); }
00128
00129 bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00130 unsigned unmap_flags) throw() override
00131 { return ALLOC::release(cap, task, unmap_flags); }
00132
00133 void operator delete(void *) {}
00134 };
00135
00136 inline
00137 Cap_alloc::~Cap_alloc()
00138 {}
00139
00140 extern Cap_alloc *virt_cap_alloc;
00141
00146 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00147 class Smart_cap_auto
00148 {
00149 private:
00150 Cap_alloc *_ca;
00151
00152 public:
00153 Smart_cap_auto() : _ca(0) {}
00154 Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}
00155
00156 void free(L4::Cap_base &c)
00157 {
00158 if (c.is_valid() && _ca)
00159 _ca->free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00160
00161 invalidate(c);
00162 }
00163
00164 static void invalidate(L4::Cap_base &c)

```

```

00165 {
00166 if (c.is_valid())
00167 c.invalidate();
00168 }
00169
00170 static L4::Cap_base copy(L4::Cap_base const &src)
00171 {
00172 L4::Cap_base r = src;
00173 invalidate(const_cast<L4::Cap_base &>(src));
00174 return r;
00175 }
00176 };
00177
00181 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00182 class Smart_count_cap
00183 {
00184 private:
00185 Cap_alloc *_ca;
00186
00187 public:
00188 Smart_count_cap() : _ca(nullptr) {}
00189 Smart_count_cap(Cap_alloc *ca) : _ca(ca) {}
00194 void free(L4::Cap_base &c) throw()
00195 {
00196 if (c.is_valid())
00197 {
00198 if (_ca && _ca->release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00199 c.invalidate();
00200 }
00201 }
00202
00206 static void invalidate(L4::Cap_base &c) throw()
00207 {
00208 if (c.is_valid())
00209 c.invalidate();
00210 }
00211
00215 L4::Cap_base copy(L4::Cap_base const &src)
00216 {
00217 if (src.is_valid())
00218 _ca->take(L4::Cap<void>(src.cap()));
00219 return src;
00220 }
00221 };
00222
00248 template< typename T >
00249 struct Auto_cap
00250 {
00251 typedef L4::Smart_cap<T, Smart_cap_auto<> > Cap;
00252 } L4_DEPRECATED("use L4::Re::Unique_cap");
00253
00284 template< typename T >
00285 struct Auto_del_cap
00286 {
00287 typedef L4::Smart_cap<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
00288 > Cap;
00288 } L4_DEPRECATED("use L4::Re::Unique_del_cap");
00291 }

```

## 15.173 l4/re/util/cap\_alloc File Reference

Capability allocator.

```

#include <l4/re/util/cap_alloc_impl.h>
#include <l4/sys/smart_capability>
#include <l4/sys/task>
#include <l4/re/consts>

```

The graph consists of numerous nodes, each containing a mathematical expression. The nodes are interconnected by directed edges, forming a complex network. The expressions involve variables  $x$ ,  $y$ , and  $z$ , and parameters  $\alpha$ ,  $\beta$ , and  $\gamma$ . The graph shows a hierarchical structure with many connections between nodes, including self-loops and bidirectional edges. The nodes are arranged in a roughly circular pattern with a central cluster of nodes.

[illegible]

- class `L4Re::Util::Smart_cap_auto< Unmap_flags >`  
*Helper for `Auto_cap` and `Auto_del_cap`.*
- class `L4Re::Util::Smart_count_cap< Unmap_flags >`  
*Helper for `Ref_cap` and `Ref_del_cap`.*
- struct `L4Re::Util::Auto_cap< T >`  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct `L4Re::Util::Auto_del_cap< T >`  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*
- struct `L4Re::Util::Ref_cap< T >`  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct `L4Re::Util::Ref_del_cap< T >`  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*



## Namespaces

- [L4Re](#)  
*[L4Re C++ Interfaces.](#)*
- [L4Re::Util](#)  
*Documentation of the [L4 Runtime Environment](#) utility functionality in C++.*

## Functions

- `template<typename T >  
Auto_cap< T >::Cap L4Re::Util::make\_auto\_cap ()`  
*Allocate a capability slot and wrap it in an [Auto\\_cap](#).*
- `template<typename T >  
Auto_del_cap< T >::Cap L4Re::Util::make\_auto\_del\_cap ()`  
*Allocate a capability slot and wrap it in an [Auto\\_del\\_cap](#).*
- `template<typename T >  
Ref_cap< T >::Cap L4Re::Util::make\_ref\_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_cap](#).*
- `template<typename T >  
Ref_del_cap< T >::Cap L4Re::Util::make\_ref\_del\_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).*

## Variables

- `_Cap_alloc` & [L4Re::Util::cap\\_alloc](#)  
*Capability allocator.*

### 15.173.1 Detailed Description

Capability allocator.

Definition in file [cap\\_alloc](#).

## 15.174 cap\_alloc

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005 * Alexander Warg <warg@os.inf.tu-dresden.de>
00006 * economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021
```

```

00026 #pragma once
00027
00028 #include <l4/re/util/cap_alloc_impl.h>
00029 #include <l4/sys/smart_capability>
00030 #include <l4/sys/task>
00031 #include <l4/re/consts>
00032
00033 namespace L4Re { namespace Util {
00034
00053 extern _Cap_alloc &cap_alloc;
00054
00058 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00059 class Smart_cap_auto
00060 {
00061 public:
00065 static void free(L4::Cap_base &c)
00066 {
00067 if (c.is_valid())
00068 {
00069 cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00070 c.invalidate();
00071 }
00072 }
00073
00077 static void invalidate(L4::Cap_base &c)
00078 {
00079 if (c.is_valid())
00080 c.invalidate();
00081 }
00082
00086 static L4::Cap_base copy(L4::Cap_base const &src)
00087 {
00088 L4::Cap_base r = src;
00089 invalidate(const_cast<L4::Cap_base &>(src));
00090 return r;
00091 }
00092 };
00093
00094
00098 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00099 class Smart_count_cap
00100 {
00101 public:
00106 static void free(L4::Cap_base &c) throw()
00107 {
00108 if (c.is_valid())
00109 {
00110 if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00111 c.invalidate();
00112 }
00113 }
00114
00118 static void invalidate(L4::Cap_base &c) throw()
00119 {
00120 if (c.is_valid())
00121 c.invalidate();
00122 }
00123
00127 static L4::Cap_base copy(L4::Cap_base const &src)
00128 {
00129 cap_alloc.take(L4::Cap<void>(src.cap()));
00130 return src;
00131 }
00132 };
00133
00134
00162 template< typename T >
00163 struct Auto_cap
00164 {
00165 typedef L4::Smart_cap<T, Smart_cap_auto< L4_FP_ALL_SPACES>
00166 > Cap;
00167 L4_DEPRECATED("use L4Re::Util::Unique_cap");
00167
00199 template< typename T >
00200 struct Auto_del_cap
00201 {
00202 typedef L4::Smart_cap<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
00203 > Cap;
00204 L4_DEPRECATED("use L4Re::Util::Unique_cap");
00204
00234 template< typename T >
00235 struct Ref_cap
00236 {
00237 typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES>
00238 > Cap;
00238 };
00239

```

```

00275 template< typename T >
00276 struct Ref_del_cap
00277 {
00278 typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ>
00279 > Cap;
00279 };
00280
00288 #pragma GCC diagnostic push
00289 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00290 template< typename T >
00291 typename Auto_cap<T>::Cap
00292 L4_DEPRECATED("use make_unique_cap")
00293 make_auto_cap()
00294 { return typename Auto_cap<T>::Cap(cap_alloc.alloc<T>()); }
00295
00303 template< typename T >
00304 typename Auto_del_cap<T>::Cap
00305 make_auto_del_cap()
00306 { return typename Auto_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00307 #pragma GCC diagnostic pop
00308
00314 template< typename T >
00315 typename Ref_cap<T>::Cap
00316 make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }
00317
00323 template< typename T >
00324 typename Ref_del_cap<T>::Cap
00325 make_ref_del_cap()
00326 { return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00327
00330 }}
00331

```

## 15.175 l4/re/consts File Reference

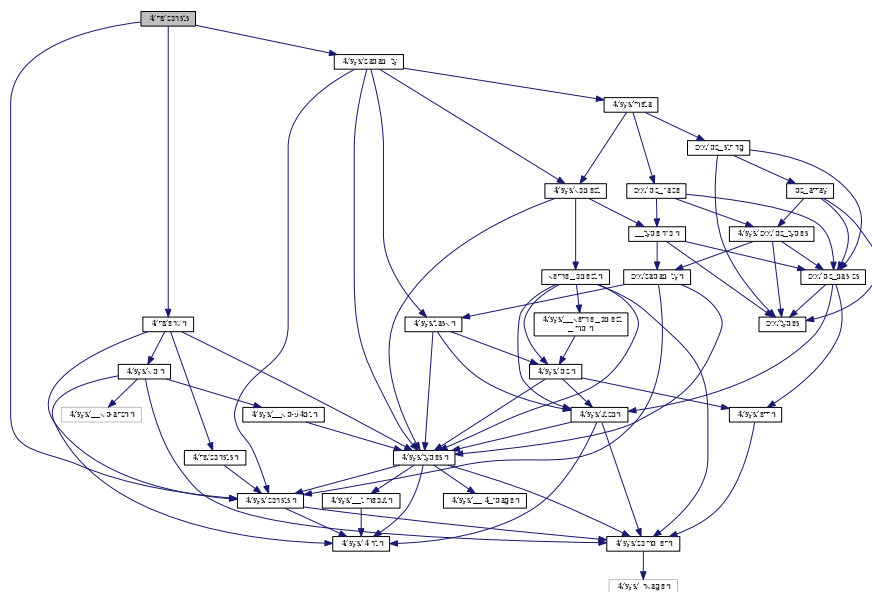
Constants.

```

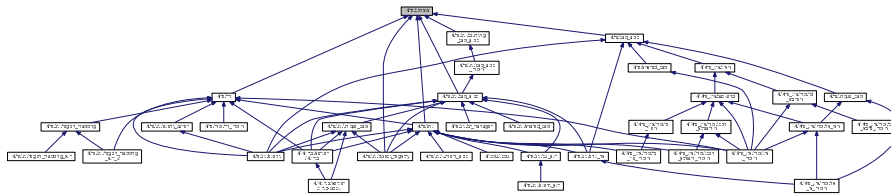
#include <l4/sys/capability>
#include <l4/sys/consts.h>
#include <l4/re/env.h>

```

Include dependency graph for consts:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.175.1 Detailed Description

Constants.

Definition in file [consts](#).

### 15.176 consts

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/sys/consts.h>
00028 #include <l4/re/env.h>
00029
00030 namespace L4Re {
00031 static L4::Cap<L4::Task>::Cap_type const This_task
00032 = (L4::Cap<L4::Task>::Cap_type) (L4RE_THIS_TASK_CAP);
00033 }

```



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.177.1 Detailed Description

Dataspace interface.

Definition in file [dataspace](#).

## 15.178 dataspace

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027
00028 #pragma once
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/l4int.h>
00032 #include <l4/sys/capability>
00033 #include <l4/re/protocols.h>
00034 #include <l4/sys/cxx/ipc_types>
00035 #include <l4/sys/cxx/ipc_iface>
00036
00037 namespace L4Re
00038 {
00039
00040 // MISSING:
00041 // * size support in map, mapped size in reply
00042
00059 class L4_EXPORT Dataspace :
00060 public L4::Kobject_t<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00061 L4::Type_info::Demand_t<1> >
00062 {
00063 public:
00064
00068 enum Map_flags
00069 {
00070 Map_ro = 0,
00071 Map_rw = 1,
00072
00073 Map_normal = 0x00,
00074 Map_cacheable = Map_normal,
00075 Map_bufferable = 0x10,
00076 Map_uncacheable = 0x20,
00077
00078 Map_caching_mask = 0x30,
00079 Map_caching_shift = 4,
00080 };
00081
00085 struct Stats {
00086 unsigned long size;
00087 unsigned long flags;

```

```

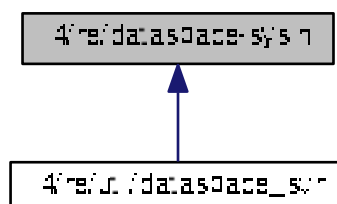
00088 };
00089
00090
00113 long map(l4_addr_t offset, unsigned long flags, l4_addr_t local_addr,
00114 l4_addr_t min_addr, l4_addr_t max_addr) const throw();
00115
00131 long map_region(l4_addr_t offset, unsigned long flags,
00132 l4_addr_t min_addr, l4_addr_t max_addr) const throw();
00133
00151 L4_RPC(long, clear, (l4_addr_t offset, unsigned long size));
00152
00172 L4_RPC(long, allocate, (l4_addr_t offset, l4_size_t size));
00173
00191 L4_RPC(long, copy_in, (l4_addr_t dst_offs,
L4::IpC::Cap<Dataspace> src,
00192 l4_addr_t src_offs, unsigned long size));
00193
00194
00212 L4_RPC(long, phys, (l4_addr_t offset, l4_addr_t &phys_addr,
l4_size_t &phys_size));
00213
00219 unsigned long size() const throw();
00220
00229 long flags() const throw();
00230
00239 L4_RPC(long, info, (Stats *stats));
00240
00250 L4_RPC(long, take, ());
00251
00261 L4_RPC(long, release, ());
00262
00263 L4_RPC_NF(long, map, (unsigned long offset, l4_addr_t spot,
00264 unsigned long flags, L4::IpC::Rcv_fpage r,
00265 L4::IpC::Snd_fpage &fp));
00266 private:
00267
00268 long __map(l4_addr_t offset, unsigned char *size, unsigned long flags,
00269 l4_addr_t local_addr) const throw();
00270
00271 public:
00272 typedef L4::Typeid::Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00273 release_t, phys_t, allocate_t> Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00274 release_t, phys_t, allocate_t> Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00275 release_t, phys_t, allocate_t> Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00276 release_t, phys_t, allocate_t> Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00277 release_t, phys_t, allocate_t> Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00278 release_t, phys_t, allocate_t> Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00279 release_t, phys_t, allocate_t> Rpc<map_t, clear_t, info_t, copy_in_t, take_t,

```

## 15.179 l4/re/dataspace-sys.h File Reference

Dataspace protocol definition.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*

### 15.179.1 Detailed Description

Dataspace protocol defintion.

Definition in file [dataspace-sys.h](#).

## 15.180 dataspace-sys.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Dataspace_
00028 {
00034 enum Opcodes { Map, Clear, Stats, Copy, Take, Release, Phys, Allocate };
00035 };
00036 };
00037

```

### 15.181 l4/re/debug File Reference

Debug interface.

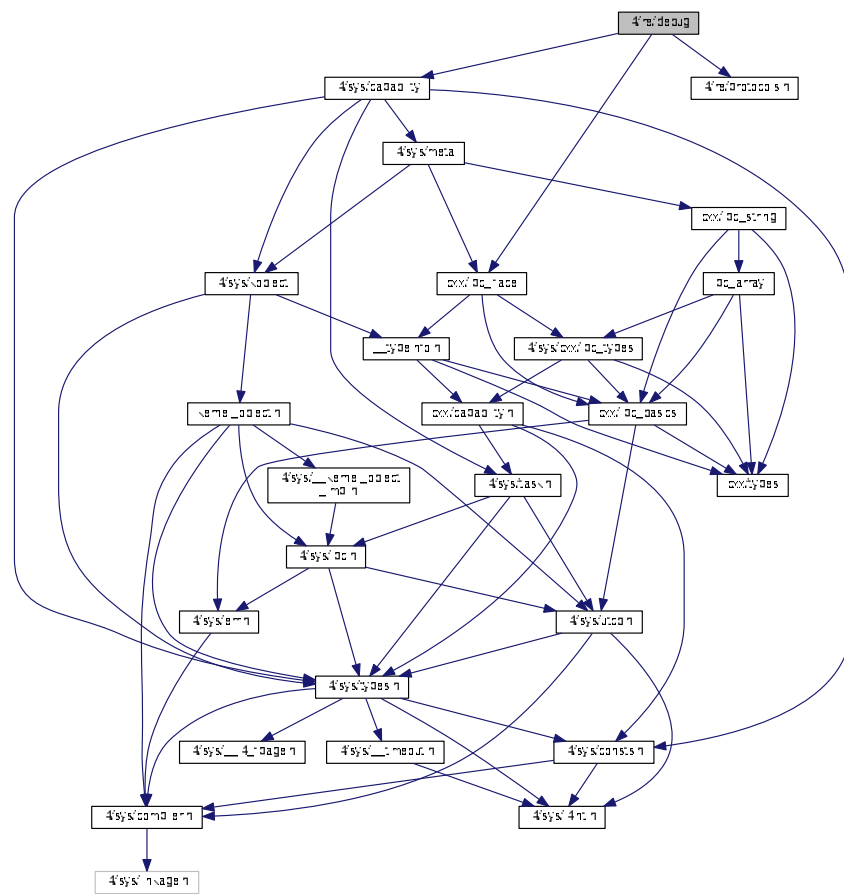
```

#include <l4/sys/capability>
#include <l4/re/protocols.h>

```



```
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for debug:
```



## Data Structures

- class [L4Re::Debug\\_obj](#)  
*Debug interface.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## 15.181.1 Detailed Description

Debug interface.

Definition in file [debug](#).

## 15.182 debug

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/re/protocols.h>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4Re {
00051 class L4_EXPORT Debug_obj :
00052 public L4::Kobject_t<Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG>
00053 {
00054 public:
00055
00063 L4_INLINE_RPC(long, debug, (unsigned long function));
00064 typedef L4::Typeid::Rpc_nocode<debug_t> Rpc;
00065 };
00066
00067 template<typename BASE>
00068 class Debug_obj_t :
00069 public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00070 {
00071 typedef L4::Typeid::Rpc<> Rpc;
00072 };
00073 }

```

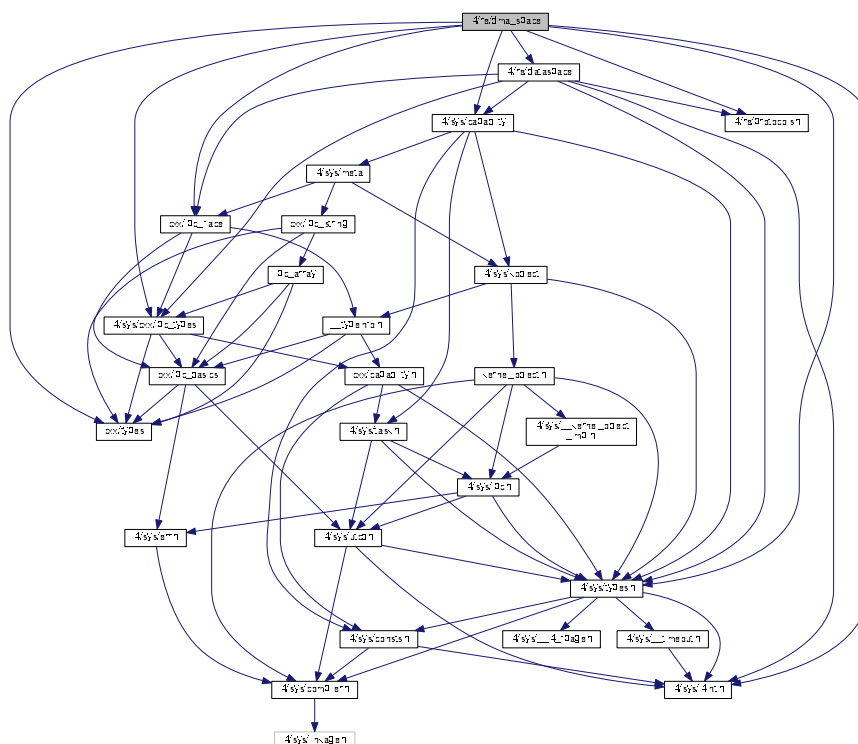
## 15.183 l4/re/dma\_space File Reference

```

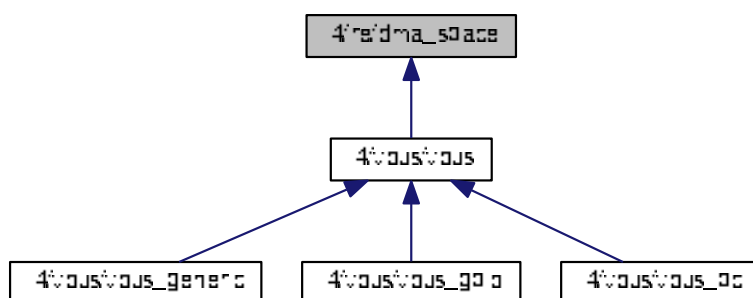
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/dataspace>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/types>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for dma\_space:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Dma_space`  
*DMA Address Space.*

## Namespaces

- **L4Re**  
*L4Re C++ Interfaces.*

## 15.184 dma\_space

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00006 /*
00007 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/capability>
00028 #include <l4/re/dataspace>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/types>
00031 #include <l4/sys/cxx/ipc_types>
00032 #include <l4/sys/cxx/ipc_iface>
00033
00034 namespace L4Re
00035 {
00036
00061 class Dma_space :
00062 public L4::Kobject_0t< Dma_space,
00063 L4RE_PROTO_DMA_SPACE,
00064 L4::Type_info::Demand_t<1> >
00065 {
00066 public:
00068 typedef l4_uint64_t Dma_addr;
00069
00073 enum Direction
00074 {
00075 Bidirectional,
00076 To_device,
00077 From_device,
00078 None
00079 };
00080
00085 enum Attribute
00086 {
00098 No_sync
00099 };
00100
00106 typedef L4::Types::Flags<Attribute> Attributes;
00107
00113 enum Space_attrib
00114 {
00121 Coherent,
00122
00127 Phys_space
00128 };
00129
00131 typedef L4::Types::Flags<Space_attrib>
Space_attribs;
00132
00157 L4_INLINE_RPC(
00158 long, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
l4_addr_t offset,
00159 L4::Ipc::In_out<l4_size_t *> size,
00160 Attributes attrs, Direction dir,
00161 Dma_addr *dma_addr));
00162
00171 L4_INLINE_RPC(
00172 long, unmap, (Dma_addr dma_addr,
00173 l4_size_t size, Attributes attrs, Direction dir));
00174
00186 L4_INLINE_RPC(
00187 long, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> >
dma_task,
00188 Space_attribs attr),
00189 L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00190
00196 L4_INLINE_RPC(

```

```

00197 long, disassociate, (),
00198 L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00199
00200 typedef L4::Typeid::Rpc<map_t, unmap_t, associate_t, disassociate_t>
00201 Rpc;
00202 };
00203 }
00204

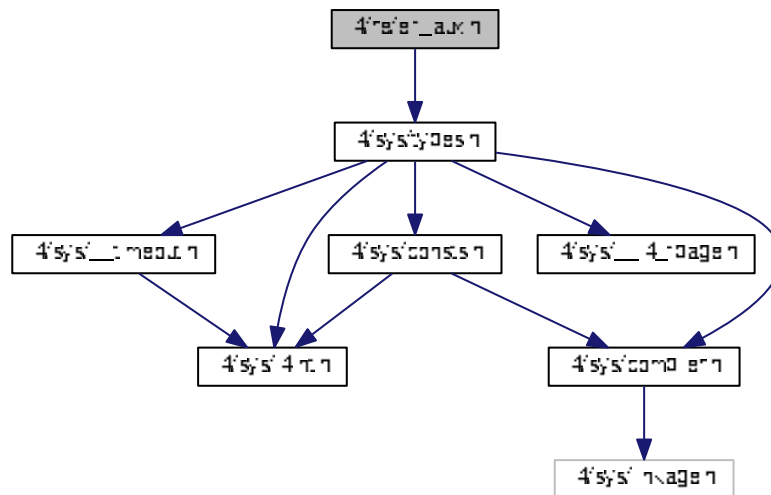
```

## 15.185 l4/re/elf\_aux.h File Reference

Auxiliary information for binaries.

```
#include <l4/sys/types.h>
```

Include dependency graph for elf\_aux.h:



## Data Structures

- struct [l4re\\_elf\\_aux\\_t](#)  
Generic header for each auxiliary vector element.
- struct [l4re\\_elf\\_aux\\_vma\\_t](#)  
Auxiliary vector element for a reserved virtual memory area.
- struct [l4re\\_elf\\_aux\\_mword\\_t](#)  
Auxiliary vector element for a single unsigned data word.

## Macros

- #define [L4RE\\_ELF\\_AUX\\_ELEM](#) const \_\_attribute\_\_((used, section(".ro.l4re\_elf\_aux"), aligned(sizeof([l4re\\_elf\\_aux\\_mword\\_t](#)))))  
Define an auxiliary vector element.
- #define [L4RE\\_ELF\\_AUX\\_ELEM\\_T](#)(type, id, tag, val...) static [L4RE\\_ELF\\_AUX\\_ELEM](#) type id = {tag, sizeof(type), val}  
Define an auxiliary vector element.

## Typedefs

- typedef struct [l4re\\_elf\\_aux\\_t](#) [l4re\\_elf\\_aux\\_t](#)  
*Generic header for each auxiliary vector element.*
- typedef struct [l4re\\_elf\\_aux\\_vma\\_t](#) [l4re\\_elf\\_aux\\_vma\\_t](#)  
*Auxiliary vector element for a reserved virtual memory area.*
- typedef struct [l4re\\_elf\\_aux\\_mword\\_t](#) [l4re\\_elf\\_aux\\_mword\\_t](#)  
*Auxiliary vector element for a single unsigned data word.*

## Enumerations

- enum {  
    [L4RE\\_ELF\\_AUX\\_T\\_NONE](#) = 0, [L4RE\\_ELF\\_AUX\\_T\\_VMA](#), [L4RE\\_ELF\\_AUX\\_T\\_STACK\\_SIZE](#), [L4RE\\_ELF\\_AUX\\_T\\_STACK\\_ADDR](#),  
    [L4RE\\_ELF\\_AUX\\_T\\_KIP\\_ADDR](#) }

### 15.185.1 Detailed Description

Auxiliary information for binaries.

Definition in file [elf\\_aux.h](#).

### 15.186 elf\_aux.h

```

00001
00005 /*
00006 * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025
00026
00039
00052 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro.l4re_elf_aux"),
 aligned(sizeof(l4_umword_t))))
00053
00067 #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) \
00068 static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
00069
00070 enum
00071 {
00075 L4RE_ELF_AUX_T_NONE = 0,
00076
00080 L4RE_ELF_AUX_T_VMA,
00081
00086 L4RE_ELF_AUX_T_STACK_SIZE,
00087
00092 L4RE_ELF_AUX_T_STACK_ADDR,
00093
00098 L4RE_ELF_AUX_T_KIP_ADDR,
00099 };

```

```

00100
00104 typedef struct l4re_elf_aux_t
00105 {
00106 l4_umword_t type;
00107 l4_umword_t length;
00108 } l4re_elf_aux_t;
00109
00113 typedef struct l4re_elf_aux_vma_t
00114 {
00115 l4_umword_t type;
00116 l4_umword_t length;
00117 l4_umword_t start;
00118 l4_umword_t end;
00119 } l4re_elf_aux_vma_t;
00120
00124 typedef struct l4re_elf_aux_mword_t
00125 {
00126 l4_umword_t type;
00127 l4_umword_t length;
00128 l4_umword_t value;
00129 } l4re_elf_aux_mword_t;
00130

```

## 15.187 l4/re/env File Reference

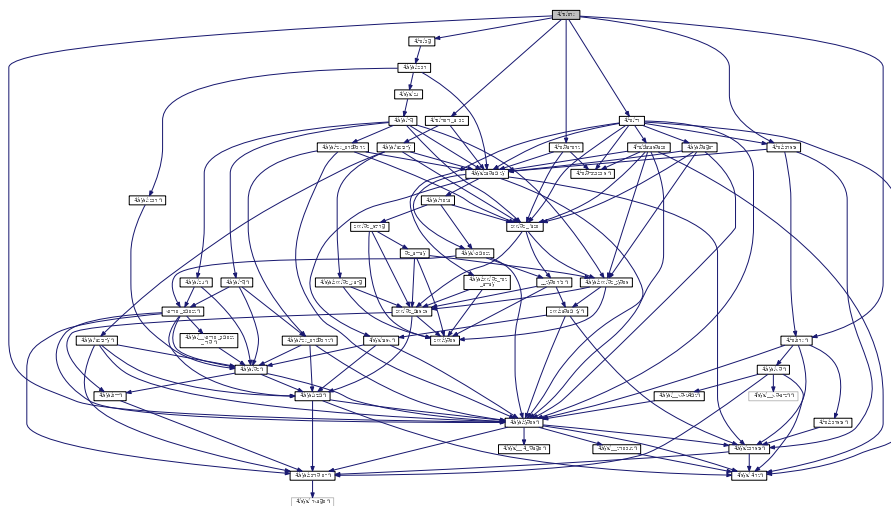
Environment interface.

```

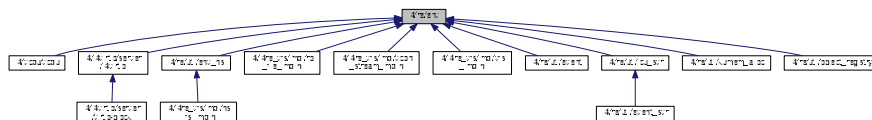
#include <l4/sys/types.h>
#include <l4/re/rm>
#include <l4/re/parent>
#include <l4/re/mem_alloc>
#include <l4/re/log>
#include <l4/re/consts>
#include <l4/re/env.h>

```

Include dependency graph for env:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Env](#)

*C++ interface of the initial environment that is provided to an [L4](#) task.*

## Namespaces

- [L4](#)  
*[L4](#) low-level kernel interface.*
- [L4Re](#)  
*[L4Re](#) C++ Interfaces.*

### 15.187.1 Detailed Description

Environment interface.

Definition in file [env](#).

## 15.188 env

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/types.h>
00029
00030 #include <l4/re/rm>
00031 #include <l4/re/parent>
00032 #include <l4/re/mem_alloc>
00033 #include <l4/re/log>
00034 #include <l4/re/consts>
00035
00036 #include <l4/re/env.h>
00037
00038 namespace L4 {
00039 class Scheduler;
00040 }
00041
00042 namespace L4Re
00043 {
00044 {
00085 class L4_EXPORT Env
00086 {
00087 private:
00088 l4re_env_t _env;
00089 public:
00090
00094 typedef l4re_env_cap_entry_t Cap_entry;
00095
00103 static Env const *env() throw()

```



```

00104 { return reinterpret_cast<Env*>(l4re_global_env); }
00105
00110 L4::Cap<Parent> parent() const throw()
00111 { return L4::Cap<Parent>(_env.parent); }
00116 L4::Cap<Mem_alloc> mem_alloc() const throw()
00117 { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
00121 L4::Cap<L4::Factory> user_factory() const throw()
00122 { return L4::Cap<L4::Factory>(_env.mem_alloc); }
00127 L4::Cap<Rm> rm() const throw()
00128 { return L4::Cap<Rm>(_env.rm); }
00133 L4::Cap<Log> log() const throw()
00134 { return L4::Cap<Log>(_env.log); }
00139 L4::Cap<L4::Thread> main_thread() const throw()
00140 { return L4::Cap<L4::Thread>(_env.main_thread); }
00145 L4::Cap<L4::Task> task() const throw()
00146 { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
00151 L4::Cap<L4::Factory> factory() const throw()
00152 { return L4::Cap<L4::Factory>(_env.factory); }
00159 l4_cap_idx_t first_free_cap() const throw()
00160 { return _env.first_free_cap; }
00165 l4_fpage_t utcb_area() const throw()
00166 { return _env.utcb_area; }
00174 l4_addr_t first_free_utcb() const throw()
00175 { return _env.first_free_utcb; }
00176
00181 Cap_entry const *initial_caps() const throw()
00182 { return _env.caps; }
00183
00192 Cap_entry const *get(char const *name, unsigned l) const throw()
00193 { return l4re_env_get_cap_l(name, l, &_env); }
00194
00203 template< typename T >
00204 L4::Cap<T> get_cap(char const *name, unsigned l) const throw()
00205 {
00206 if (Cap_entry const *e = get(name, l))
00207 return L4::Cap<T>(e->cap);
00208
00209 return L4::Cap<T>(-L4_ENOENT);
00210 }
00211
00218 template< typename T >
00219 L4::Cap<T> get_cap(char const *name) const throw()
00220 { return get_cap<T>(name, __builtin_strlen(name)); }
00221
00226 void parent(L4::Cap<Parent> const &c) throw()
00227 { _env.parent = c.cap(); }
00232 void mem_alloc(L4::Cap<Mem_alloc> const &c) throw()
00233 { _env.mem_alloc = c.cap(); }
00238 void rm(L4::Cap<Rm> const &c) throw()
00239 { _env.rm = c.cap(); }
00244 void log(L4::Cap<Log> const &c) throw()
00245 { _env.log = c.cap(); }
00250 void main_thread(L4::Cap<L4::Thread> const &c) throw()
00251 { _env.main_thread = c.cap(); }
00256 void factory(L4::Cap<L4::Factory> const &c) throw()
00257 { _env.factory = c.cap(); }
00262 void first_free_cap(l4_cap_idx_t c) throw()
00263 { _env.first_free_cap = c; }
00268 void utcb_area(l4_fpage_t utcbs) throw()
00269 { _env.utcb_area = utcbs; }
00274 void first_free_utcb(l4_addr_t u) throw()
00275 { _env.first_free_utcb = u; }
00276
00282 L4::Cap<L4::Scheduler> scheduler() const throw()
00283 { return L4::Cap<L4::Scheduler>(_env.scheduler); }
00284
00289 void scheduler(L4::Cap<L4::Scheduler> const &c) throw()
00290 { _env.scheduler = c.cap(); }
00291
00296 void initial_caps(Cap_entry *first) throw()
00297 { _env.caps = first; }
00298 };
00299 };

```

## 15.189 l4/re/env.h File Reference

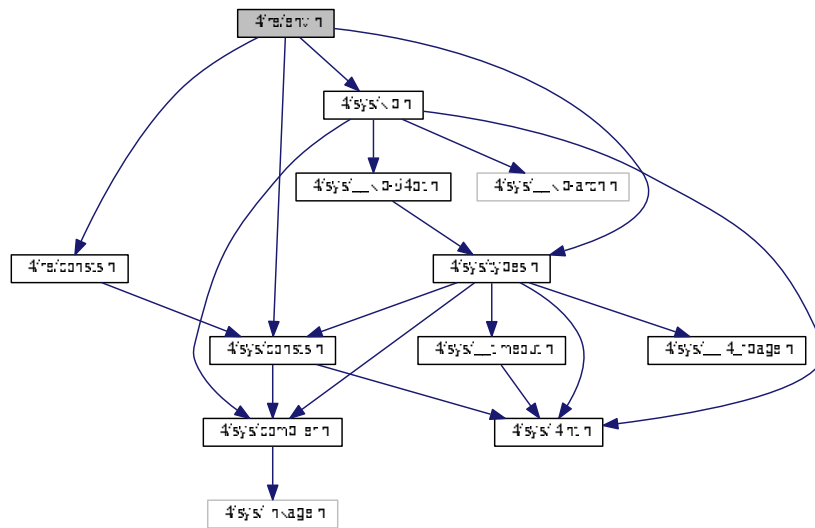
Environment interface.

```

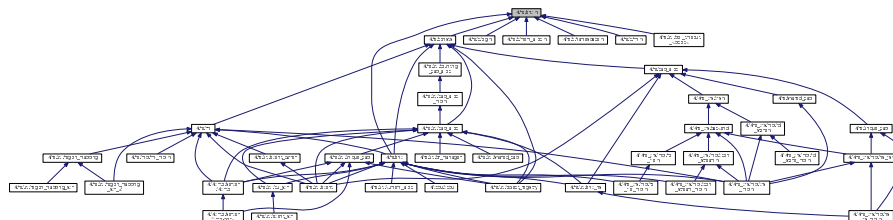
#include <l4/sys/consts.h>
#include <l4/sys/types.h>

```

```
#include <l4/sys/kip.h>
#include <l4/re/consts.h>
Include dependency graph for env.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4re_env_cap_entry_t`  
Entry in the `L4Re` environment array for the named initial objects.
- struct `l4re_env_t`  
Initial environment data structure.

## Typedefs

- typedef struct `l4re_env_cap_entry_t` `l4re_env_cap_entry_t`  
Entry in the `L4Re` environment array for the named initial objects.
- typedef struct `l4re_env_t` `l4re_env_t`  
Initial environment data structure.

## Functions

- `l4re_env_t * l4re_env (void)` `L4_NOTHROW`  
*Get L4Re initial environment.*
- `l4_kernel_info_t * l4re_kip (void)` `L4_NOTHROW`  
*Get Kernel Info Page.*
- `l4_cap_idx_t l4re_env_get_cap (char const *name)` `L4_NOTHROW`  
*Get the capability selector for the object named name.*
- `l4_cap_idx_t l4re_env_get_cap_e (char const *name, l4re_env_t const *e)` `L4_NOTHROW`  
*Get the capability selector for the object named name.*
- `l4re_env_cap_entry_t const * l4re_env_get_cap_l (char const *name, unsigned l, l4re_env_t const *e)` `L4_NOTHROW`  
*Get the full l4re\_env\_cap\_entry\_t for the object named name.*

### 15.189.1 Detailed Description

Environment interface.

Definition in file [env.h](#).

### 15.189.2 Typedef Documentation

#### 15.189.2.1 l4re\_env\_t

```
typedef struct l4re_env_t l4re_env_t
```

Initial environment data structure.

See also

[Initial environment](#)

## 15.190 env.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
```

```

00023 #pragma once
00024
00025 #include <l4/sys/consts.h>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/kip.h>
00028
00029 #include <l4/re/consts.h>
00030
00049 typedef struct l4re_env_cap_entry_t
00050 {
00054 l4_cap_idx_t cap;
00055
00060 l4_umword_t flags;
00061
00065 char name[16];
00066 #ifdef __cplusplus
00067
00071 l4re_env_cap_entry_t() : cap(L4_INVALID_CAP), flags(~0) {}
00072
00080 l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c,
00081 l4_umword_t f = 0)
00082 : cap(c), flags(f)
00083 {
00084 for (unsigned i = 0; n && i < sizeof(name); ++i, ++n)
00085 {
00085 name[i] = *n;
00086 if (!*n)
00087 break;
00088 }
00089 }
00090
00091 static bool is_valid_name(char const *n)
00092 {
00093 for (unsigned i = 0; *n; ++i, ++n)
00094 if (i > sizeof(name))
00095 return false;
00096
00097 return true;
00098 }
00099 #endif
00100 } l4re_env_cap_entry_t;
00101
00102
00108 typedef struct l4re_env_t
00109 {
00110 l4_cap_idx_t parent;
00111 l4_cap_idx_t rm;
00112 l4_cap_idx_t mem_alloc;
00113 l4_cap_idx_t log;
00114 l4_cap_idx_t main_thread;
00115 l4_cap_idx_t factory;
00116 l4_cap_idx_t scheduler;
00117 l4_cap_idx_t first_free_cap;
00118 l4_fpage_t utcb_area;
00119 l4_addr_t first_free_utcb;
00120 l4re_env_cap_entry_t *caps;
00121 } l4re_env_t;
00122
00128 extern l4re_env_t *l4re_global_env;
00129
00130
00136 L4_INLINE l4re_env_t *l4re_env(void) L4_NOTHROW;
00137
00138 /*
00139 * FIXME: this seems to be at the wrong place here
00140 */
00146 L4_INLINE l4_kernel_info_t *l4re_kip(void) L4_NOTHROW;
00147
00148
00156 L4_INLINE l4_cap_idx_t
00157 l4re_env_get_cap(char const *name) L4_NOTHROW;
00158
00167 L4_INLINE l4_cap_idx_t
00168 l4re_env_get_cap_e(char const *name, l4re_env_t const *e)
00169 L4_NOTHROW;
00170
00180 L4_INLINE l4re_env_cap_entry_t const *
00181 l4re_env_get_cap_l(char const *name, unsigned l,
00182 l4re_env_t const *e) L4_NOTHROW;
00183
00183 L4_INLINE
00184 l4re_env_t *l4re_env() L4_NOTHROW
00185 { return l4re_global_env; }
00186
00187 L4_INLINE
00188 l4_kernel_info_t *l4re_kip() L4_NOTHROW
00189 {

```

```

00190 extern char __L4_KIP_ADDR__[];
00191 return (l4_kernel_info_t *)__L4_KIP_ADDR__;
00192 }
00193
00194 L4_INLINE l4re_env_cap_entry_t const *
00195 l4re_env_get_cap_l(char const *name, unsigned l,
00196 l4re_env_t const *e) L4_NOTHROW
00197 {
00198 l4re_env_cap_entry_t const *c = e->caps;
00199 for (; c && c->flags != ~0UL; ++c)
00200 {
00201 unsigned i;
00202 for (i = 0;
00203 i < sizeof(c->name) && i < l && c->name[i] && name[i] &&
00204 name[i] == c->name[i];
00205 ++i)
00206 if (i == l && (i == sizeof(c->name) || !c->name[i]))
00207 return c;
00208 }
00209 return NULL;
00210 }
00211
00212 L4_INLINE l4_cap_idx_t
00213 l4re_env_get_cap_e(char const *name, l4re_env_t const *e)
00214 L4_NOTHROW
00215 {
00216 unsigned l;
00217 l4re_env_cap_entry_t const *r;
00218 for (l = 0; name[l]; ++l) ;
00219 r = l4re_env_get_cap_l(name, l, e);
00220 if (r)
00221 return r->cap;
00222 return L4_INVALID_CAP;
00223 }
00224
00225 L4_INLINE l4_cap_idx_t
00226 l4re_env_get_cap(char const *name) L4_NOTHROW
00227 { return l4re_env_get_cap_e(name, l4re_env()); }
00228

```

## 15.191 l4/re/error\_helper File Reference

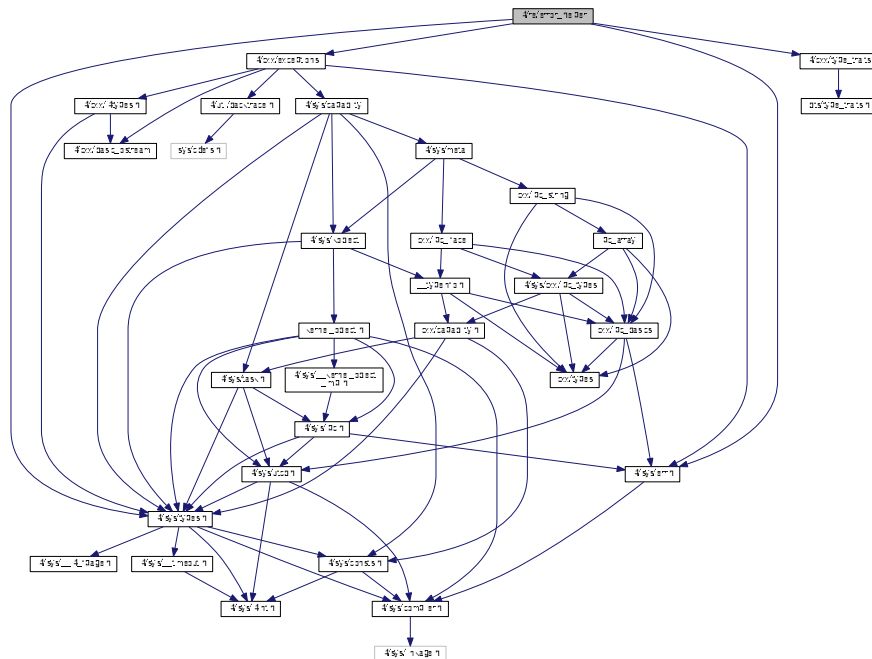
Error helper.

```

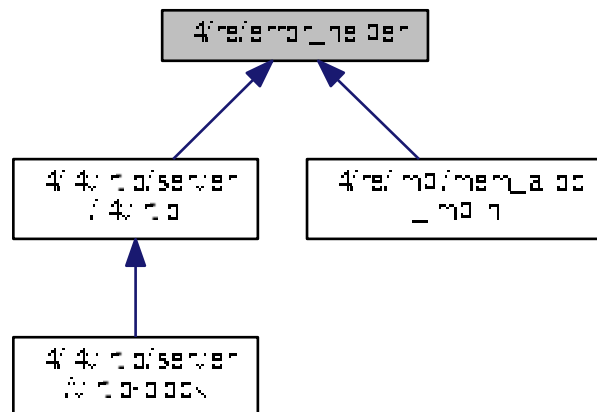
#include <l4/sys/types.h>
#include <l4/cxx/exceptions>
#include <l4/cxx/type_traits>
#include <l4/sys/err.h>

```

Include dependency graph for `error_helper`:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Functions

- long `L4Re::chksys` (long err, char const \*extra="", long ret=0)  
*Generate C++ exception on error.*
- long `L4Re::chksys` (`l4_msgtag_t` const &t, char const \*extra="", `l4_utcb_t` \*utcb=`l4_utcb()`, long ret=0)  
*Generate C++ exception on error.*
- long `L4Re::chksys` (`l4_msgtag_t` const &t, `l4_utcb_t` \*utcb, char const \*extra="")  
*Generate C++ exception on error.*
- template<typename T >  
T `L4Re::chkcap` (T &&cap, char const \*extra="", long err=-`L4_ENOMEM`)  
*Check for valid capability or raise C++ exception.*
- `l4_msgtag_t` `L4Re::chkipc` (`l4_msgtag_t` tag, char const \*extra="", `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Test a message tag for IPC errors.*

### 15.191.1 Detailed Description

Error helper.

Definition in file [error\\_helper](#).

## 15.192 error\_helper

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/cxx/exceptions>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/sys/err.h>
00031
00032 namespace L4Re {
00033
00034 #ifdef __EXCEPTIONS
00035 namespace Priv {
00036 inline long __attribute__((__noreturn__)) __runtime_error(long err, char const *extra);
00037
00038 inline long __runtime_error(long err, char const *extra)
00039 {
00040 switch (err)
00041 {
00042 case -L4_ENOENT: throw (L4::Element_not_found(extra));
00043 case -L4_ENOMEM: throw (L4::Out_of_memory(extra));
00044 case -L4_EEXIST: throw (L4::Element_already_exists(extra));
00045 case -L4_ERANGE: throw (L4::Bounds_error(extra));
00046 default: throw (L4::Runtime_error(err, extra));
00047 }
00048 }
00049
00050 }
```

```

00051
00062 inline
00063 long chksys(long err, char const *extra = "", long ret = 0)
00064 {
00065 if (L4_UNLIKELY(err < 0))
00066 Priv::__runtime_error(ret ? ret : err, extra);
00067 return err;
00068 }
00069
00070
00083 inline
00084 long chksys(l4_msgtag_t const &t, char const *extra = "",
00085 l4_utcb_t *utcb = l4_utcb(), long ret = 0)
00086 {
00087 if (L4_UNLIKELY(t.has_error()))
00088 Priv::__runtime_error(ret ? ret : l4_error_u(t, utcb), extra);
00089 else if (L4_UNLIKELY(t.label() < 0))
00090 throw L4::Runtime_error(ret ? ret : t.label(), extra);
00091 return t.label();
00092 }
00093
00094
00106 inline
00107 long chksys(l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra = "")
00108 { return chksys(t, extra, utcb); }
00109
00110 #if 0
00111 inline
00112 long chksys(long ret, long err, char const *extra = "")
00113 {
00114 if (L4_UNLIKELY(ret < 0))
00115 Priv::__runtime_error(err, extra);
00116 return ret;
00117 }
00118 #endif
00119
00120
00137 template<typename T>
00138 inline
00139 #if __cplusplus >= 201103L
00140 T chkcap(T &&cap, char const *extra = "", long err = -L4_ENOMEM)
00141 #else
00142 T chkcap(T cap, char const *extra = "", long err = -L4_ENOMEM)
00143 #endif
00144 {
00145 if (L4_UNLIKELY(!cap.is_valid()))
00146 Priv::__runtime_error(err ? err : cap.cap(), extra);
00147
00148 #if __cplusplus >= 201103L
00149 return cxx::forward<T>(cap);
00150 #else
00151 return cap;
00152 #endif
00153 }
00154
00169 inline
00170 l4_msgtag_t
00171 chkipc(l4_msgtag_t tag, char const *extra = "",
00172 l4_utcb_t *utcb = l4_utcb())
00173 {
00174 if (L4_UNLIKELY(tag.has_error()))
00175 chksys(l4_error_u(tag, utcb), extra);
00176 return tag;
00177 }
00178
00179 #endif
00180
00181 }

```

## 15.193 l4/re/util/event File Reference

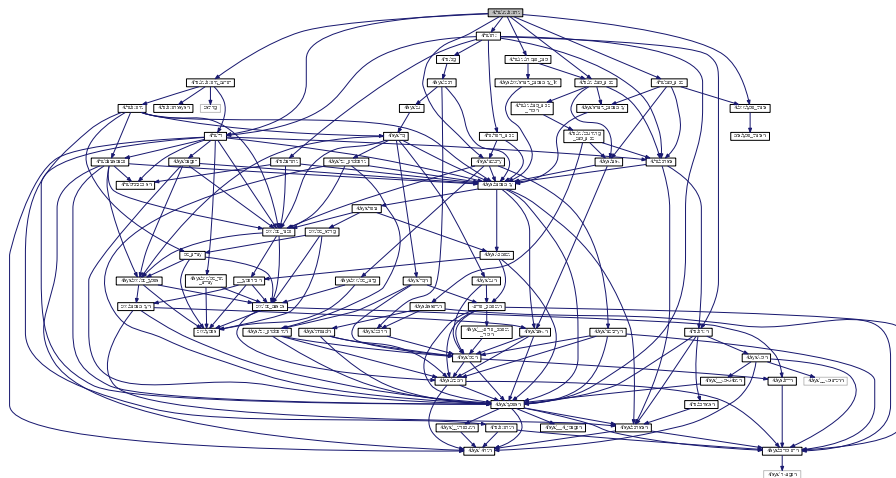
```

#include <l4/re/cap_alloc>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/unique_cap>
#include <l4/re/env>
#include <l4/re/rm>
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>

```



```
#include <l4/cxx/type_traits>
Include dependency graph for event:
```



## Data Structures

- class [L4Re::Util::Event\\_t< PAYLOAD >](#)  
*Convenience wrapper for getting access to an event object.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

## 15.194 event

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/re/util/cap_alloc>
00027 #include <l4/re/util/unique_cap>
00028 #include <l4/re/env>
00029 #include <l4/re/rm>
```

```

00030 #include <l4/re/util/event_buffer>
00031 #include <l4/sys/factory>
00032 #include <l4/cxx/type_traits>
00033
00034 namespace L4Re { namespace Util {
00035
00042 template< typename PAYLOAD >
00043 class Event_t
00044 {
00045 public:
00049 enum Mode
00050 {
00051 Mode_irq,
00052 Mode_polling,
00053 };
00054
00069 template<typename IRQ_TYPE>
00070 int init(L4::Cap<L4Re::Event> event,
00071 L4Re::Env const *env = L4Re::Env::env(),
00072 L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc
(L4Re::Util::cap_alloc))
00073 {
00074 Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<
L4Re::Dataspace>());
00075 if (!ev_ds.is_valid())
00076 return -L4_ENOMEM;
00077
00078 int r;
00079
00080 Unique_del_cap<IRQ_TYPE> ev_irq(ca->alloc<IRQ_TYPE>());
00081 if (!ev_irq.is_valid())
00082 return -L4_ENOMEM;
00083
00084 if ((r = l4_error(env->factory()->create(ev_irq.get()))))
00085 return r;
00086
00087 if ((r = l4_error(event->bind(0, ev_irq.get()))))
00088 return r;
00089
00090 if ((r = event->get_buffer(ev_ds.get())))
00091 return r;
00092
00093 long sz = ev_ds->size();
00094 if (sz < 0)
00095 return sz;
00096
00097 Rm::Unique_region<void*> buf;
00098
00099 if ((r = env->rm()->attach(&buf, sz, L4Re::Rm::Search_addr,
00100 L4::Ipc::make_cap_rw(ev_ds.get()))))
00101 return r;
00102
00103 _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00104 _ev_ds = cxx::move(ev_ds);
00105 _ev_irq = cxx::move(ev_irq);
00106 _buf = cxx::move(buf);
00107
00108 return 0;
00109 }
00110
00122 int init_poll(L4::Cap<L4Re::Event> event,
00123 L4Re::Env const *env = L4Re::Env::env(),
00124 L4Re::Cap_alloc *ca =
L4Re::Cap_alloc::get_cap_alloc(
L4Re::Util::cap_alloc))
00125 {
00126 Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<
L4Re::Dataspace>());
00127 if (!ev_ds.is_valid())
00128 return -L4_ENOMEM;
00129
00130 int r;
00131
00132 if ((r = event->get_buffer(ev_ds.get())))
00133 return r;
00134
00135 long sz = ev_ds->size();
00136 if (sz < 0)
00137 return sz;
00138
00139 Rm::Unique_region<void*> buf;
00140
00141 if ((r = env->rm()->attach(&buf, sz, L4Re::Rm::Search_addr,
00142 L4::Ipc::make_cap_rw(ev_ds.get()))))
00143 return r;
00144
00145 _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);

```

## 15.195 I4/re/impl/dataspace\_impl.h File Reference

```
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
Include dependency graph for dataspace_impl.h:
```



- Generated for L4Re by Doxygen

### 15.195.1 Detailed Description

Dataspace client stub implementation.

Definition in file [dataspace\\_impl.h](#).

## 15.196 dataspace\_impl.h

```

00001
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #include <l4/re/dataspace>
00021 #include <l4/sys/cxx/ipc_client>
00022
00023 L4_RPC_DEF(L4Re::Dataspace::clear);
00024 L4_RPC_DEF(L4Re::Dataspace::allocate);
00025 L4_RPC_DEF(L4Re::Dataspace::copy_in);
00026 L4_RPC_DEF(L4Re::Dataspace::phys);
00027 L4_RPC_DEF(L4Re::Dataspace::info);
00028 L4_RPC_DEF(L4Re::Dataspace::take);
00029 L4_RPC_DEF(L4Re::Dataspace::release);
00030
00031 namespace L4Re {
00032
00033 long
00034 Dataspace::__map(unsigned long offset, unsigned char *size, unsigned long flags,
00035 l4_addr_t local_addr) const throw()
00036 {
00037 l4_addr_t spot = local_addr & ~(~0UL << l4_umword_t(*size));
00038 l4_addr_t base = local_addr & (~0UL << l4_umword_t(*size));
00039 L4::Ipc::Rcv_fpage r;
00040 r = L4::Ipc::Rcv_fpage::mem(base, *size, 0);
00041 L4::Ipc::Snd_fpage fp;
00042 long err = map_t::call(c(), offset, spot, flags, r, fp, l4_utcb());
00043 if (L4_UNLIKELY(err < 0))
00044 return err;
00045 *size = fp.rcv_order();
00046 return err;
00047 }
00048
00049 long
00050 Dataspace::map_region(l4_addr_t offset, unsigned long flags,
00051 l4_addr_t min_addr, l4_addr_t max_addr) const throw()
00052 {
00053 min_addr = l4_trunc_page(min_addr);
00054 max_addr = l4_round_page(max_addr);
00055 unsigned char order = L4_LOG2_PAGESIZE;
00056
00057 long err = 0;
00058
00059 while (min_addr < max_addr)
00060 {
00061 unsigned char order_mapped;
00062 order_mapped = order
00063 = l4_fpage_max_order(order, min_addr, min_addr, max_addr, min_addr);
00064 err = __map(offset, &order_mapped, flags, min_addr);
00065 if (L4_UNLIKELY(err < 0))
00066 return err;
00067 if (order > order_mapped)
00068 order = order_mapped;
00069 }
00070 }

```

```

00075
00076 min_addr += 1UL << order;
00077 offset += 1UL << order;
00078
00079 if (min_addr >= max_addr)
00080 return 0;
00081
00082 while (min_addr != l4_trunc_size(min_addr, order)
00083 || max_addr < l4_round_size(min_addr + 1, order))
00084 --order;
00085 }
00086
00087 return 0;
00088 }
00089
00090
00091 long
00092 Dataspace::map(l4_addr_t offset, unsigned long flags,
00093 l4_addr_t local_addr,
00094 l4_addr_t min_addr, l4_addr_t max_addr) const throw()
00095 {
00096 min_addr = l4_trunc_page(min_addr);
00097 max_addr = l4_round_page(max_addr);
00098 local_addr = l4_trunc_page(local_addr);
00099 unsigned char order
00100 = l4_fpage_max_order(L4_LOG2_PAGESIZE, local_addr, min_addr, max_addr
00101 , local_addr);
00102
00103 return __map(offset, &order, flags, local_addr);
00104 }
00105
00106 unsigned long
00107 Dataspace::size() const throw()
00108 {
00109 Stats stats = Stats();
00110 int err = info(&stats);
00111 if (err < 0)
00112 return 0;
00113 return stats.size;
00114 }
00115
00116 long
00117 Dataspace::flags() const throw()
00118 {
00119 Stats stats = Stats();
00120 int err = info(&stats);
00121 if (err < 0)
00122 return err;
00123 return stats.flags;
00124 }
00125 };

```

## 15.197 l4/re/impl/mem\_alloc\_impl.h File Reference

Memory allocator client stub implementation.

```

#include <l4/re/mem_alloc>
#include <l4/re/mem_alloc-sys.h>
#include <l4/re/dataspace>
#include <l4/re/error_helper>
#include <l4/sys/factory>

```



```

00027
00028 #include <l4/sys/factory>
00029
00030
00031 namespace L4Re
00032 {
00033
00034 long
00035 Mem_alloc::alloc(long size,
00036 L4::Cap<Dataspace> mem, unsigned long flags,
00037 unsigned long align) const throw()
00038 {
00039 L4::Cap<L4::Factory> f(cap());
00040 return l4_error(f->create(mem, L4Re::Dataspace::Protocol)
00041 << l4_mword_t(size)
00042 << l4_umword_t(flags)
00043 << l4_umword_t(align));
00044 }
00045
00046 long
00047 Mem_alloc::free(L4::Cap<Dataspace>) const throw()
00048 {
00049 return 0;
00050 }
00051
00052 };

```

## 15.199 l4/re/impl/namespace\_impl.h File Reference

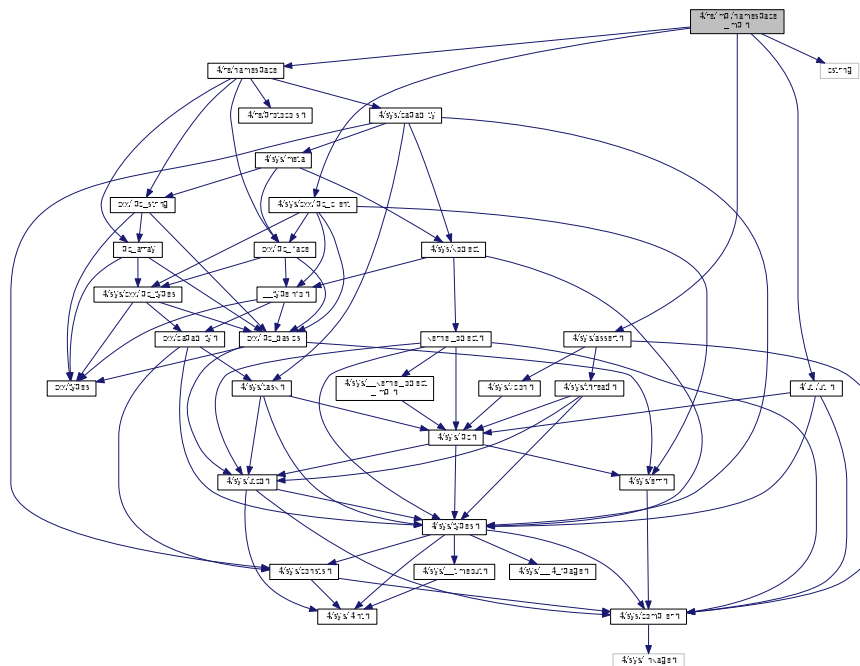
Namespace client stub implementation.

```

#include <l4/re/namespace>
#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>
#include <cstring>

```

Include dependency graph for namespace\_impl.h:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.199.1 Detailed Description

Namespace client stub implementation.

Definition in file [namespace\\_impl.h](#).

### 15.200 namespace\_impl.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #include <l4/re/namespace>
00024
00025 #include <l4/util/util.h>
00026 #include <l4/sys/cxx/ipc_client>
00027 #include <l4/sys/assert.h>
00028
00029 #include <cstring>
00030
00031 L4_RPC_DEF(L4Re::Namespace::query);
00032 L4_RPC_DEF(L4Re::Namespace::register_obj);
00033 L4_RPC_DEF(L4Re::Namespace::unlink);
00034
00035 namespace L4Re {
00036
00037 long
00038 Namespace::_query(char const *name, unsigned len,
00039 L4::Cap<void> const &target,
00040 l4_umword_t *local_id, bool iterate) const throw()
00041 {
00042 l4_assert(target.is_valid());
00043
00044 L4::Cap<Namespace> ns = c();
00045 L4::Ipc::Array<char const, unsigned long> _name(len, name);
00046
00047 while (_name.length > 0)
00048 {
00049 L4::Ipc::Snd_fpage cap;
00050 L4::Opcode dummy;
00051 int err = query_t::call(ns, _name,
00052 L4::Ipc::Small_buf(target.cap(),
00053 local_id
00054 ? L4_RCV_ITEM_LOCAL_ID
00055 : 0),
00056 cap, dummy, _name);
00057 if (err < 0)
00058 return err;
00059
00060 bool const partly = err & Partly_resolved;
00061 if (cap.id_received())
00062 {
00063 *local_id = cap.data();
00064 return _name.length;

```



```

00065 }
00066
00067 if (partly && iterate)
00068 ns = L4::cap_cast<Namespace>(target);
00069 else
00070 return err;
00071 }
00072
00073 return _name.length;
00074 }
00075
00076 long
00077 Namespace::query(char const *name, unsigned len, L4::Cap<void> const &target,
00078 int timeout, l4_umword_t *local_id, bool iterate) const throw()
00079 {
00080 if (L4_UNLIKELY(len == 0))
00081 return -L4_EINVAL;
00082
00083 long ret;
00084 long rem = timeout;
00085 long to = 0;
00086
00087 if (rem)
00088 to = 10;
00089 do
00090 {
00091 ret = _query(name, len, target, local_id, iterate);
00092
00093 if (ret >= 0)
00094 return ret;
00095
00096 if (L4_UNLIKELY(ret < 0 && (ret != -L4_EAGAIN)))
00097 return ret;
00098
00099 if (rem == to)
00100 return ret;
00101
00102 l4_sleep(to);
00103
00104 if (rem > 0)
00105 {
00106 rem -= to;
00107 if (to > rem)
00108 to = rem;
00109 }
00110
00111 if (to < 100)
00112 to += to;
00113 }
00114 while (486);
00115 }
00116
00117 long
00118 Namespace::query(char const *name, L4::Cap<void> const &target,
00119 int timeout, l4_umword_t *local_id,
00120 bool iterate) const throw()
00121 {
00122 return query(name, __builtin_strlen(name), target,
00123 timeout, local_id, iterate);
00124 }
00125
00126 }

```

## 15.201 l4/re/impl/rm\_impl.h File Reference

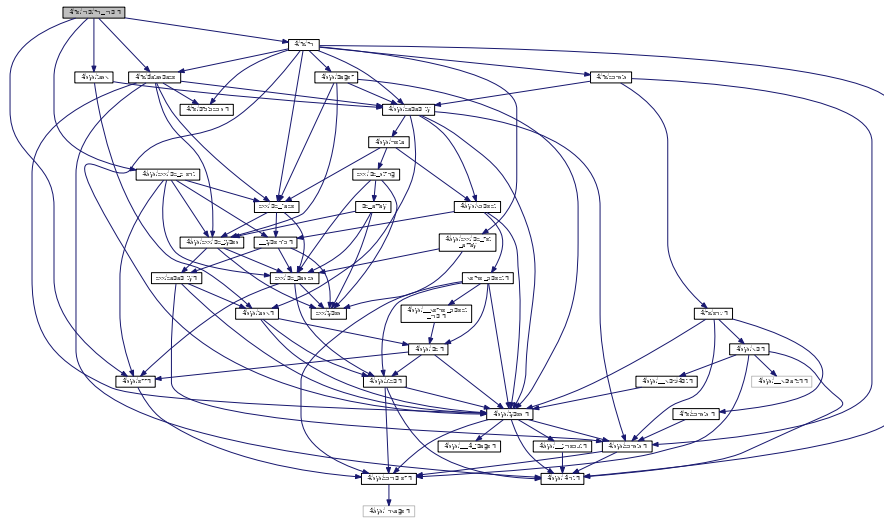
Region map client stub implementation.

```

#include <l4/re/rm>
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/task>
#include <l4/sys/err.h>

```

Include dependency graph for `rm_impl.h`:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.201.1 Detailed Description

Region map client stub implementation.

Definition in file [rm\\_impl.h](#).

### 15.202 rm\_impl.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #include <l4/re/rm>
00024 #include <l4/re/dataspace>
00025
00026 #include <l4/sys/cxx/ipc_client>
00027
00028 #include <l4/sys/task>
00029 #include <l4/sys/err.h>
00030

```

```

00031 L4_RPC_DEF(L4Re::Rm::reserve_area);
00032 L4_RPC_DEF(L4Re::Rm::free_area);
00033 L4_RPC_DEF(L4Re::Rm::attach);
00034 L4_RPC_DEF(L4Re::Rm::detach);
00035 L4_RPC_DEF(L4Re::Rm::get_regions);
00036 L4_RPC_DEF(L4Re::Rm::get_areas);
00037 L4_RPC_DEF(L4Re::Rm::find);
00038
00039 namespace L4Re
00040 {
00041
00042 long
00043 Rm::attach(l4_addr_t *start, unsigned long size, unsigned long flags,
00044 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs,
00045 unsigned char align) const throw()
00046 {
00047 if (flags & Reserved)
00048 mem = L4::Ipc::Cap<L4Re::Dataspace>();
00049
00050 long e = attach_t::call(c(), start, size, flags, mem, offs, align, mem.cap().cap());
00051 if (e < 0)
00052 return e;
00053
00054 if (flags & Eager_map)
00055 {
00056 unsigned long fl = (flags & Read_only)
00057 ? Dataspace::Map_ro
00058 : Dataspace::Map_rw;
00059 fl |= (flags & Caching) >> Caching_ds_shift;
00060 e = mem.cap()->map_region(offs, fl, *start, *start + size);
00061 }
00062 return e;
00063 }
00064
00065 int
00066 Rm::detach(l4_addr_t start, unsigned long size,
00067 L4::Cap<Dataspace> *mem,
00068 L4::Cap<L4::Task> task, unsigned flags) const throw()
00069 {
00070 l4_addr_t rstart = 0, rsize = 0;
00071 l4_cap_idx_t mem_cap = L4_INVALID_CAP;
00072 long e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
00073 if (L4_UNLIKELY(e < 0))
00074 return e;
00075
00076 if (mem)
00077 *mem = L4::Cap<L4Re::Dataspace>(mem_cap);
00078
00079 if (!task.is_valid())
00080 return e;
00081
00082 rsize = l4_round_page(rsize);
00083 unsigned order = L4_LOG2_PAGESIZE;
00084 unsigned long sz = (1UL << order);
00085 for (unsigned long p = rstart; rsize; p += sz, rsize -= sz)
00086 {
00087 while (sz > rsize)
00088 {
00089 --order;
00090 sz >>= 1;
00091 }
00092
00093 for (;;)
00094 {
00095 unsigned long m = sz << 1;
00096 if (m > rsize)
00097 break;
00098
00099 if (p & (m - 1))
00100 break;
00101
00102 ++order;
00103 sz <<= 1;
00104 }
00105
00106 task->unmap(l4_fpage(p, order, L4_FPAGE_RWX),
00107 L4_FP_ALL_SPACES);
00108 }
00109 return e;
00110 }
00111 }

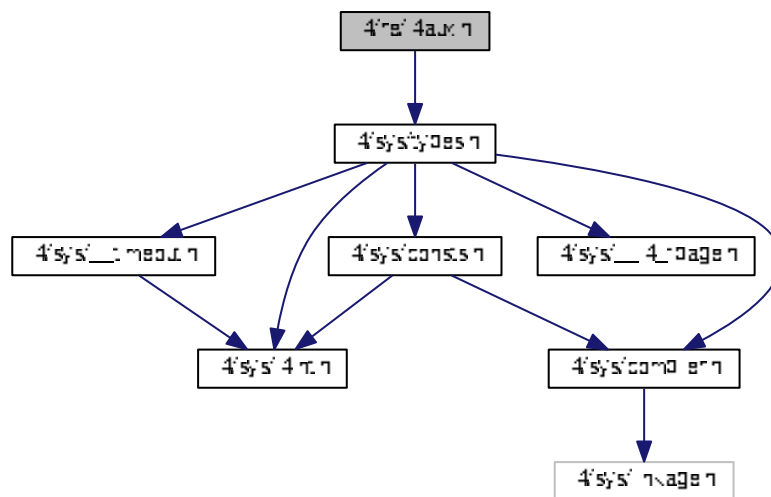
```

## 15.203 l4/re/l4aux.h File Reference

Auxiliary definitions.

```
#include <l4/sys/types.h>
```

Include dependency graph for l4aux.h:



### Data Structures

- struct [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Typedefs

- typedef struct [l4re\\_aux\\_t](#) [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Enumerations

- enum [l4re\\_aux\\_ldr\\_flags\\_t](#)  
*Flags for program loading.*

#### 15.203.1 Detailed Description

Auxiliary definitions.

Definition in file [l4aux.h](#).

## 15.204 l4aux.h

```

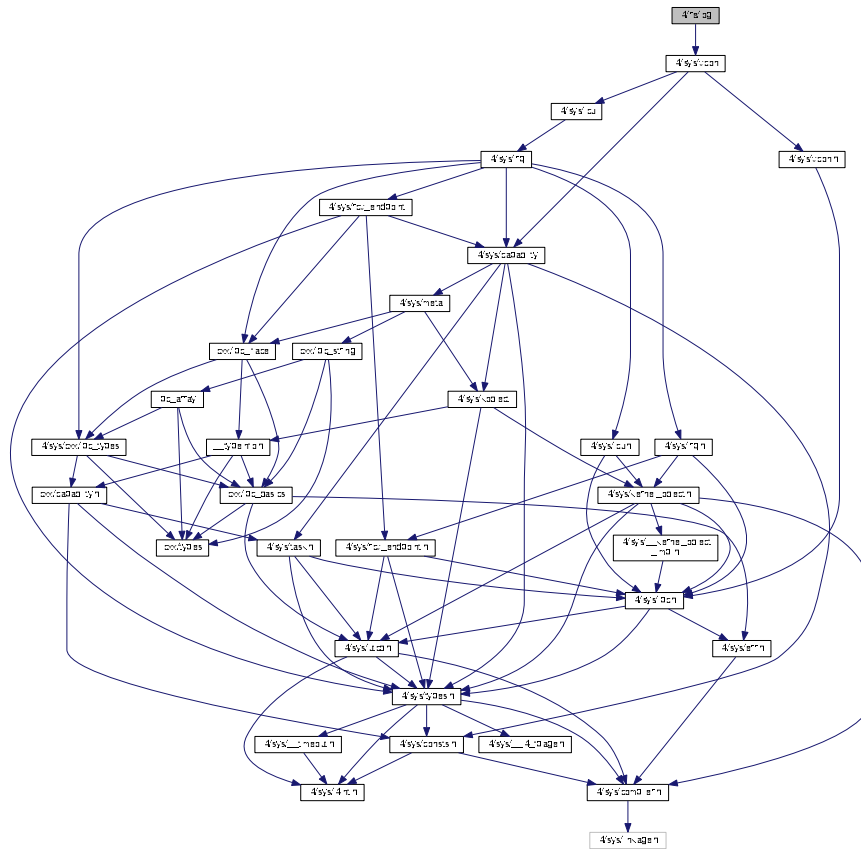
00001 #pragma once
00002
00003 /*
00004 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00006 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #include <l4/sys/types.h>
00024
00025 enum l4re_aux_ldr_flags_t
00026 {
00027 L4RE_AUX_LDR_FLAG_EAGER_MAP = 0x1,
00028 L4RE_AUX_LDR_FLAG_ALL_SEGS_COW = 0x2,
00029 L4RE_AUX_LDR_FLAG_PINNED_SEGS = 0x4,
00030 };
00031
00032 typedef struct l4re_aux_t
00033 {
00034 char const * binary;
00035 l4_cap_idx_t kip_ds;
00036 l4_umword_t dbg_lvl;
00037 l4_umword_t ldr_flags;
00038 } l4re_aux_t;
00039

```

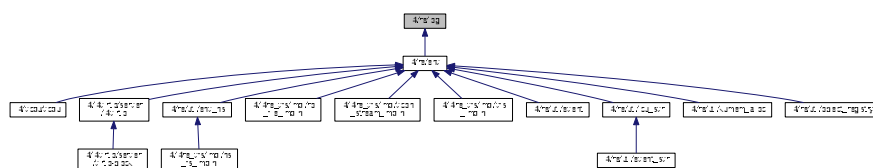
## 15.205 l4/re/log File Reference

Log interface.

```
#include <linux/sys/vcon>
Include dependency graph for log:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Log`  
*Log* interface class.

## Namespaces

- L4Re
- L4Re C++ Interfaces.*

### 15.205.1 Detailed Description

Log interface.

Definition in file [log](#).

## 15.206 log

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005 * Alexander Warg <warg@os.inf.tu-dresden.de>
00006 * economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 #include <l4/sys/vcon>
00024
00025 namespace L4Re {
00026
00027 class L4_EXPORT Log : public L4::Kobject_t<Log, L4::Vcon, L4::PROTO_EMPTY>
00028 {
00029 public:
00030 void printn(char const *string, int len) const throw();
00031 void print(char const *string) const throw();
00032 };
00033
00034 }
```

## 15.207 l4/re/log-sys.h File Reference

Log protocol definition.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Log\\_::Opcodes](#)  
*Logging-service communication-protocol opcodes.*

### 15.207.1 Detailed Description

Log protocol definition.

Definition in file [log-sys.h](#).

## 15.208 log-sys.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Log_
00028 {
00034 enum Opcodes { Print };
00035 };
00036 };
```

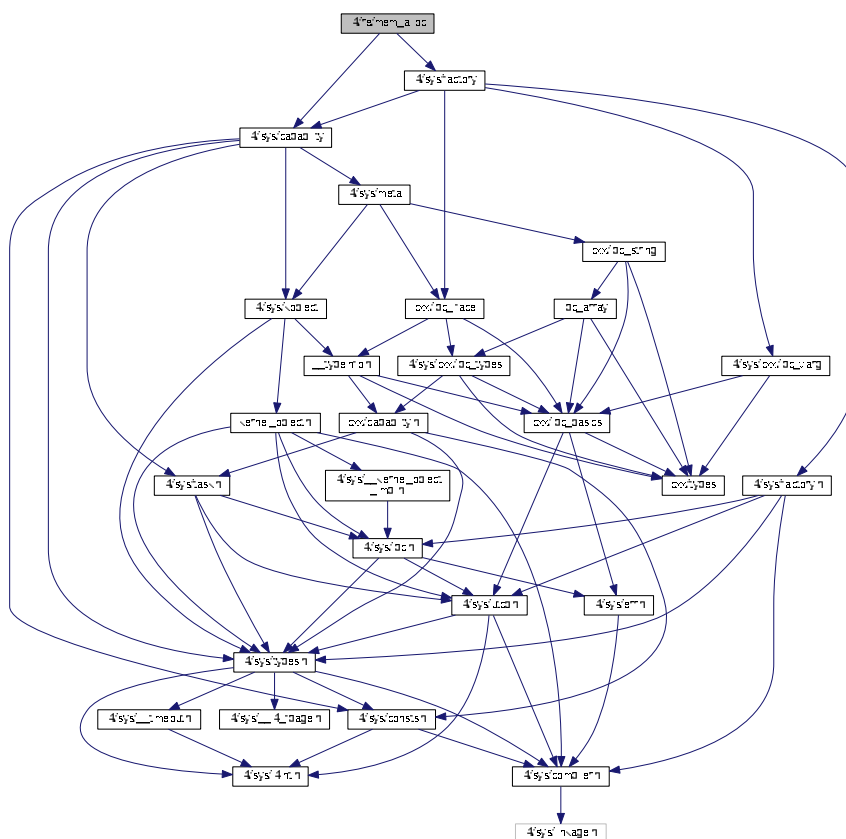
## 15.209 l4/re/mem\_alloc File Reference

Memory allocator interface.

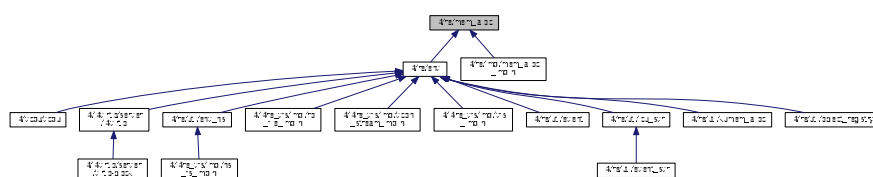
```
#include <l4/sys/capability>
#include <l4/sys/factory>
```



Include dependency graph for mem\_alloc:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Mem_alloc`  
*Memory allocation interface.*

## Namespaces

- L4Re
- L4Re C++ Interfaces.*

## 15.209.1 Detailed Description

Memory allocator interface.

Definition in file [mem\\_alloc](#).

## 15.210 mem\_alloc

```

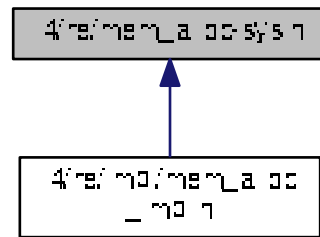
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * Copyright (C) 2015 Kernkonzept GmbH.
00005 * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006 */
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/sys/factory>
00030
00031 namespace L4Re {
00032 class Dataspace;
00033
00034 // MISSING:
00035 // * alignment constraints
00036 // * shall we support superpages in noncont memory?
00037
00038 class L4_EXPORT Mem_alloc :
00039 public L4::Kobject_t<Mem_alloc, L4::Factory, L4::PROTO_EMPTY>
00040 {
00041 public:
00042 enum Mem_alloc_flags
00043 {
00044 Continuous = 0x01,
00045 Pinned = 0x02,
00046 Super_pages = 0x04,
00047 };
00048
00049 long alloc(long size, L4::Cap<Dataspace> mem,
00050 unsigned long flags = 0, unsigned long align = 0) const throw();
00051
00052 /* Deprecation message added Q4 2016 */
00053 long free(L4::Cap<Dataspace> mem) const throw()
00054 {
00055 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only.
00056 Check documentation for details.");
00057 }
00058 };
00059
00060 };
00061
00062 };

```

## 15.211 l4/re/mem\_alloc-sys.h File Reference

Memory allocator protocol definitions.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Mem\\_alloc::\\_Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*

### 15.211.1 Detailed Description

Memory allocator protocol definitions.

Definition in file [mem\\_alloc-sys.h](#).

## 15.212 mem\_alloc-sys.h

```

00001
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021
00022 namespace L4Re
00023 {
00024 namespace Mem_alloc_
00025 {
00026 enum Opcodes { Alloc, Free };
00027 };
00028 };

```



## Data Structures

- class [L4Re::Namespace](#)  
*Name-space interface.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.213.1 Detailed Description

Namespace interface.

Definition in file [namespace](#).

## 15.214 namespace

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031 #include <l4/sys/cxx/ipc_array>
00032 #include <l4/sys/cxx/ipc_string>
00033
00034 namespace L4Re {
00035
00060 class L4_EXPORT Namespace :
00061 public L4::Kobject_t<Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE,
00062 L4::Type_info::Demand_t<1> >
00063 {
00064 public:
00068 enum Register_flags
00069 {
00070 Ro = L4_CAP_FPAGE_RO,
00071 Rw = L4_CAP_FPAGE_RW,
00072 Rs = L4_CAP_FPAGE_RS,
00073 Rws = L4_CAP_FPAGE_RWS,
00074 Strong = L4_CAP_FPAGE_S,
00075 Trusted = 0x008,
00076
00077 Cap_flags = Ro | Rw | Strong | Trusted,
00078
00079 Link = 0x100,
00080 Overwrite = 0x200,
00081 };
00082

```

```

00088 enum Query_result_flags
00089 {
00090 Partly_resolved = 0x020,
00091 };
00092
00093 enum Query_timeout
00094 {
00095 To_default = 3600000,
00096 To_non_blocking = 0,
00097 To_forever = -1,
00098 };
00099
00100 L4_RPC_NF(
00101 long, query, (L4::Ipc::Array_ref<char const, unsigned long>
name,
00102 L4::Ipc::Small_buf cap,
00103 L4::Ipc::Snd_fpage &snd_cap,
L4::Ipc::Opt<L4::Opcode &> dummy,
00104 L4::Ipc::Opt<
L4::Ipc::Array_ref<char const, unsigned long> &> out_name));
00105
00131 long query(char const *name, L4::Cap<void> const &cap,
00132 int timeout = To_default,
00133 l4_umword_t *local_id = 0, bool iterate = true) const throw();
00134
00144 long query(char const *name, unsigned len, L4::Cap<void> const &cap,
00145 int timeout = To_default,
00146 l4_umword_t *local_id = 0, bool iterate = true) const throw();
00147
00148 L4_RPC_NF(long, register_obj, (unsigned flags,
00149 L4::Ipc::Array<char const, unsigned long>
name,
00150 L4::Ipc::Opt< L4::Ipc::Cap<void> > obj),
00151 L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00152
00176 long register_obj(char const *name, L4::Ipc::Cap<void> obj,
00177 unsigned flags = Rw) const throw()
00178 {
00179 return register_obj_t::call(c(), flags,
00180 L4::Ipc::Array<char const, unsigned long>
(
00181 __builtin_strlen(name), name),
00182 obj);
00183 }
00184
00185 L4_RPC_NF_OP(3, // backward compatibility opcode
00186 long, unlink, (L4::Ipc::Array<char const, unsigned long>
name),
00187 L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00188
00202 long unlink(char const* name)
00203 {
00204 return unlink_t::call(c(), L4::Ipc::Array<char const, unsigned long>
(
00205 __builtin_strlen(name), name));
00206 }
00207
00208 typedef L4::Typeid::Rpc<query_t, register_obj_t, unlink_t>
Rpc;
00209
00210 private:
00211 long _query(char const *name, unsigned len,
00212 L4::Cap<void> const &target, l4_umword_t *local_id,
00213 bool iterate) const throw();
00214
00215 };
00216
00217 };

```

## 15.215 l4/re/namespace-sys.h File Reference

Namespace protocol definitions.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*

### 15.215.1 Detailed Description

Namespace protocol definitions.

Definition in file [namespace-sys.h](#).

## 15.216 namespace-sys.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace L4Re {
00026 namespace Namespace_
00027 {
00033 enum Opcodes { Query, Register, Link, Unlink };
00034 };
00035 };

```

### 15.217 l4/re/parent File Reference

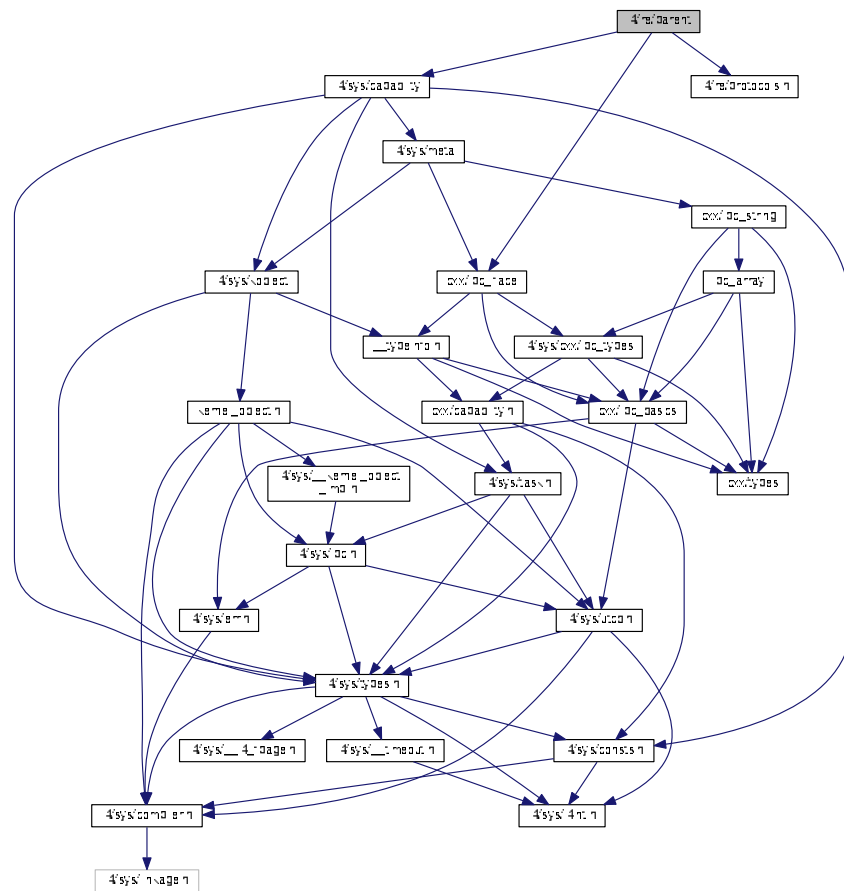
Parent interface.

```

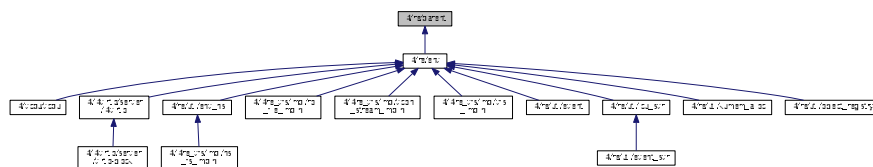
#include <l4/sys/capability>
#include <l4/re/protocols.h>

```

```
#include <linux/sys/cxx/ipc_iface>
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Parent`  
*Parent* interface.

## Namespaces

- L4Re
- L4Re C++ Interfaces.*



### 15.217.1 Detailed Description

Parent interface.

Definition in file [parent](#).

## 15.218 parent

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/re/protocols.h>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4Re {
00032
00051 class L4_EXPORT Parent :
00052 public L4::Kobject_t<Parent, L4::Kobject, L4RE_PROTO_PARENT>
00053 {
00054 public:
00064 L4_INLINE_RPC(long, signal, (unsigned long sig, unsigned long val));
00065 typedef L4::Typeid::Rpc<signal_t> Rpc;
00066 };
00067 };
00068

```

## 15.219 l4/re/parent-sys.h File Reference

Parent protocol definition.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*

### 15.219.1 Detailed Description

Parent protocol definition.

Definition in file [parent-sys.h](#).

## 15.220 parent-sys.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Parent_
00028 {
00034 enum Opcodes { Signal };
00035 };
00036 };

```

### 15.221 l4/re/protocols.h File Reference

[L4Re](#) Protocol Constants (C version)

This graph shows which files directly or indirectly include this file:



### Enumerations

- enum [L4re\\_protocols](#) {  
[L4RE\\_PROTO\\_DATASPACE](#) = 0x4000, [L4RE\\_PROTO\\_NAMESPACE](#), [L4RE\\_PROTO\\_PARENT](#), [L4RE\\_PROTO\\_GOOS](#),  
[L4RE\\_PROTO\\_RSVD\\_1](#), [L4RE\\_PROTO\\_RM](#), [L4RE\\_PROTO\\_EVENT](#), [L4RE\\_PROTO\\_INHIBITOR](#),  
[L4RE\\_PROTO\\_DMA\\_SPACE](#), [L4RE\\_PROTO\\_MMIO\\_SPACE](#), [L4RE\\_PROTO\\_DEBUG](#) = ~0x7fffL }

### 15.221.1 Detailed Description

[L4Re](#) Protocol Constants (C version)

Definition in file [protocols.h](#).

### 15.221.2 Enumeration Type Documentation

#### 15.221.2.1 L4re\_protocols

enum [L4re\\_protocols](#)

Enumerator

|                       |                                                |
|-----------------------|------------------------------------------------|
| L4RE_PROTO_DATASPACE  | ID for <a href="#">L4Re::Dataspace</a> RPCs.   |
| L4RE_PROTO_NAMESPACE  | ID for <a href="#">L4Re::Namespace</a> RPCs.   |
| L4RE_PROTO_PARENT     | ID for <a href="#">L4Re::Parent</a> RPCs.      |
| L4RE_PROTO_GOOS       | ID for <a href="#">L4Re::Video::Goos</a> RPCs. |
| L4RE_PROTO_RSVD_1     | Reserved ID.                                   |
| L4RE_PROTO_RM         | ID for <a href="#">L4Re::Rm</a> RPCs.          |
| L4RE_PROTO_EVENT      | ID for <a href="#">L4Re::Event</a> RPCs.       |
| L4RE_PROTO_INHIBITOR  | ID for <a href="#">L4Re::Inhibitor</a> RPCs.   |
| L4RE_PROTO_DMA_SPACE  | ID for <a href="#">L4Re::Dma_space</a> RPCs.   |
| L4RE_PROTO_MMIO_SPACE | ID for <a href="#">L4Re::Mmio_space</a> .      |
| L4RE_PROTO_DEBUG      | ID for debugging RPCs.                         |

Definition at line 30 of file [protocols.h](#).

## 15.222 protocols.h

```

00001
00002 /*
00003 * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018
00019 #pragma once
00020
00021 enum L4re_protocols
00022 {
00023 L4RE_PROTO_DATASPACE = 0x4000,
00024 L4RE_PROTO_NAMESPACE,

```

```

00034 L4RE_PROTO_PARENT,
00035 L4RE_PROTO_GOOS,
00036 L4RE_PROTO_RSVD_1,
00037 L4RE_PROTO_RM,
00038 L4RE_PROTO_EVENT,
00039 L4RE_PROTO_INHIBITOR,
00040 L4RE_PROTO_DMA_SPACE,
00041 L4RE_PROTO_MMIO_SPACE,
00043 L4RE_PROTO_DEBUG = ~0x7ffffL
00044 };
00045

```

## 15.223 l4/re/rm File Reference

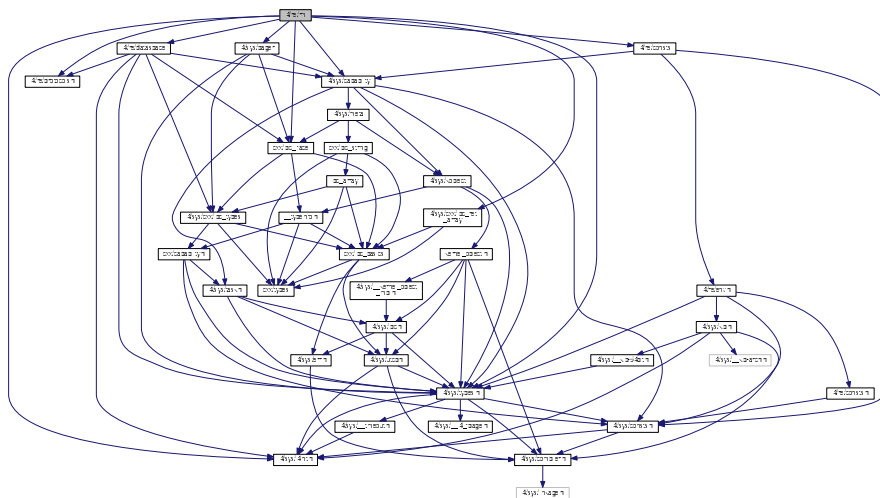
Region mapper interface.

```

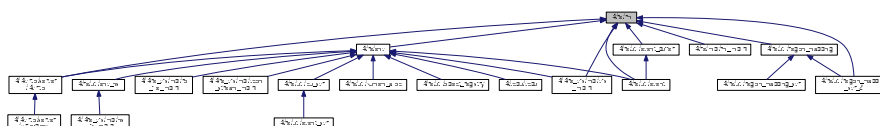
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/re/consts>
#include <l4/re/dataspace>

```

Include dependency graph for rm:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class **L4Re::Rm**  
*Region map.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.223.1 Detailed Description

Region mapper interface.

Definition in file [rm](#).

## 15.224 rm

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 #pragma once
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/l4int.h>
00031 #include <l4/sys/capability>
00032 #include <l4/re/protocols.h>
00033 #include <l4/sys/pager>
00034 #include <l4/sys/cxx/ipc_iface>
00035 #include <l4/sys/cxx/ipc_ret_array>
00036 #include <l4/re/consts>
00037 #include <l4/re/dataspace>
00038
00039 namespace L4Re {
00040
00071 class L4_EXPORT Rm :
00072 public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00073 L4::Type_info::Demand_t<1> >
00074 {
00075 public:
00077 enum Detach_result
00078 {
00079 Detached_ds = 0,
00080 Kept_ds = 1,
00081 Split_ds = 2,
00082 Detach_result_mask = 3,
00083
00084 Detach_again = 4,
00085 };
00086
00088 enum Region_flags
00089 {
00090 Read_only = 0x01,
00091 Detach_free = 0x02,
00093 Pager = 0x04,
00094 Reserved = 0x08,
00095
00097 Caching_shift = 8,
00099 Caching_ds_shift = Caching_shift - Dataspace::Map_caching_shift,
00101 Caching = Dataspace::Map_caching_mask << Caching_ds_shift,

```

```

00103 Cache_normal = Dataspace::Map_normal << Caching_ds_shift,
00105 Cache_buffered = Dataspace::Map_bufferable << Caching_ds_shift,
00107 Cache_uncached = Dataspace::Map_uncacheable << Caching_ds_shift,
00108
00109 Region_flags = Caching | 0x0f,
00110 };
00111
00113 enum Attach_flags
00114 {
00115 Search_addr = 0x20,
00116 In_area = 0x40,
00117 Eager_map = 0x80,
00118
00119 Attach_flags = 0xf0,
00120 };
00121
00123 enum Detach_flags
00124 {
00134 Detach_exact = 1,
00144 Detach_overlap = 2,
00145
00153 Detach_keep = 4,
00154 };
00155
00156
00157 template< typename T >
00158 class Auto_region
00159 {
00160 private:
00161 T _addr;
00162 mutable L4::Cap<Rm> _rm;
00163
00164 public:
00165 Auto_region() throw()
00166 : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00167
00168 explicit Auto_region(T addr) throw()
00169 : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00170
00171 Auto_region(T addr, L4::Cap<Rm> const &rm) throw()
00172 : _addr(addr), _rm(rm) {}
00173
00174 Auto_region(Auto_region const &o) throw() : _addr(o.get()), _rm(o._rm)
00175 { o.release(); }
00176
00177 Auto_region &operator = (Auto_region const &o) throw()
00178 {
00179 if (&o != this)
00180 {
00181 if (_rm.is_valid())
00182 _rm->detach(l4_addr_t(_addr), 0);
00183 _rm = o._rm;
00184 _addr = o.release();
00185 }
00186 return *this;
00187 }
00188
00189 ~Auto_region() throw()
00190 {
00191 if (_rm.is_valid())
00192 _rm->detach(l4_addr_t(_addr), 0);
00193 }
00194
00195 T get() const throw() { return _addr; }
00196 T release() const throw() { _rm = L4::Cap<Rm>::Invalid; return _addr; }
00197 void reset(T addr, L4::Cap<Rm> const &rm) throw()
00198 {
00199 if (_rm.is_valid())
00200 _rm->detach(l4_addr_t(_addr), 0);
00201
00202 _rm = rm;
00203 _addr = addr;
00204 }
00205
00206 void reset() throw()
00207 { reset(0, L4::Cap<Rm>::Invalid); }
00208
00210 T operator * () const throw() { return _addr; }
00211
00213 T operator -> () const throw() { return _addr; }
00214
00215 } L4_DEPRECATED("use L4Re::Rm::Unique_region");
00216
00241 long reserve_area(l4_addr_t *start, unsigned long size,
00242 unsigned flags = 0,
00243 unsigned char align = L4_PAGESHIFT) const throw()
00244 { return reserve_area_t::call(c(), start, size, flags, align); }

```

```

00245
00246 L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00247 unsigned long size,
00248 unsigned flags,
00249 unsigned char align));
00250
00266 template< typename T >
00267 long reserve_area(T **start, unsigned long size,
00268 unsigned flags = 0,
00269 unsigned char align = L4_PAGESHIFT) const throw()
00270 { return reserve_area_t::call(c(), (l4_addr_t*)start, size, flags, align); }
00271
00284 L4_RPC(long, free_area, (l4_addr_t addr));
00285
00286 L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00287 unsigned long size, unsigned long flags,
00288 L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
00289 l4_addr_t offs, unsigned char align,
00290 L4::Ipc::Opt<l4_cap_idx_t> client_cap));
00291
00292 L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00293 l4_addr_t &start, l4_addr_t &size,
00294 l4_cap_idx_t &mem_cap));
00295
00339 long attach(l4_addr_t *start, unsigned long size, unsigned long flags,
00340 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00341 unsigned char align = L4_PAGESHIFT) const throw();
00342
00346 template< typename T >
00347 long attach(T **start, unsigned long size, unsigned long flags,
00348 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00349 unsigned char align = L4_PAGESHIFT) const throw()
00350 {
00351 union X { l4_addr_t a; T* t; };
00352 X *x = reinterpret_cast<X*>(start);
00353 return attach(&x->a, size, flags, mem, offs, align);
00354 }
00355
00356 #pragma GCC diagnostic push
00357 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00358 template< typename T >
00359 long attach(Auto_region<T> *start, unsigned long size, unsigned long flags,
00360 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00361 unsigned char align = L4_PAGESHIFT) const throw()
00362 {
00363 l4_addr_t addr = (l4_addr_t)start->get();
00364
00365 long res = attach(&addr, size, flags, mem, offs, align);
00366 if (res < 0)
00367 return res;
00368
00369 start->reset((T)addr, L4::Cap<Rm>(cap()));
00370 return res;
00371 }
00372 #pragma GCC diagnostic pop
00373
00374 #if __cplusplus >= 201103L
00375 template< typename T >
00376 class Unique_region
00377 {
00378 private:
00379 T _addr;
00380 L4::Cap<Rm> _rm;
00381
00382 public:
00383 Unique_region(Unique_region const &) = delete;
00384 Unique_region &operator = (Unique_region const &) = delete;
00385
00386 Unique_region() noexcept
00387 : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00388
00389 explicit Unique_region(T addr) noexcept
00390 : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00391
00392 Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept
00393 : _addr(addr), _rm(rm) {}
00394
00395 Unique_region(Unique_region &o) noexcept : _addr(o.get()), _rm(o._rm)
00396 { o.release(); }
00397
00398 Unique_region &operator = (Unique_region &o) noexcept
00399 {
00400 if (&o != this)
00401 {
00402 if (_rm.is_valid())
00403 _rm->detach(l4_addr_t(_addr), 0);
00404 _rm = o._rm;

```

```

00405 _addr = o.release();
00406 }
00407 return *this;
00408 }
00409
00410 ~Unique_region() noexcept
00411 {
00412 if (_rm.is_valid())
00413 _rm->detach(l4_addr_t(_addr), 0);
00414 }
00415
00416 T get() const noexcept
00417 { return _addr; }
00418
00419 T release() noexcept
00420 {
00421 _rm = L4::Cap<Rm>::Invalid;
00422 return _addr;
00423 }
00424
00425 void reset(T addr, L4::Cap<Rm> const &rm) noexcept
00426 {
00427 if (_rm.is_valid())
00428 _rm->detach(l4_addr_t(_addr), 0);
00429
00430 _rm = rm;
00431 _addr = addr;
00432 }
00433
00434 void reset() noexcept
00435 { reset(0, L4::Cap<Rm>::Invalid); }
00436
00437 bool is_valid() const noexcept
00438 { return _rm.is_valid(); }
00439
00440 T operator * () const noexcept { return _addr; }
00441
00442 T operator -> () const noexcept { return _addr; }
00443 };
00444
00445 template< typename T >
00446 long attach(Unique_region<T> *start, unsigned long size, unsigned long flags,
00447 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00448 unsigned char align = L4_PAGESHIFT) const noexcept
00449 {
00450 l4_addr_t addr = (l4_addr_t)start->get();
00451
00452 long res = attach(&addr, size, flags, mem, offs, align);
00453 if (res < 0)
00454 return res;
00455
00456 start->reset((T)addr, L4::Cap<Rm>(cap()));
00457 return res;
00458 }
00459 #endif
00460
00461 int detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00462 L4::Cap<L4::Task> const &task = This_task) const throw();
00463
00464 int detach(void *addr, L4::Cap<Dataspace> *mem,
00465 L4::Cap<L4::Task> const &task = This_task) const throw();
00466
00467 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00468 L4::Cap<L4::Task> const &task) const throw();
00469
00470 int find(l4_addr_t *addr, unsigned long *size, l4_addr_t *offset,
00471 unsigned *flags, L4::Cap<Dataspace> *m) throw()
00472 { return find_t::call(c(), addr, size, flags, offset, m); }
00473
00474 L4_RPC_NF(int, find, (L4::Ipc::In_out<l4_addr_t *> addr,
00475 L4::Ipc::In_out<unsigned long *> size,
00476 unsigned *flags, l4_addr_t *offset,
00477 L4::Ipc::As_value<L4::Cap<Dataspace> > *m));
00478
00479 struct Region
00480 {
00481 l4_addr_t start;
00482 l4_addr_t end;
00483 l4_addr_t offset;
00484 L4::Cap<Dataspace> ds;
00485 };
00486
00487 struct Area
00488 {
00489 l4_addr_t start;
00490 l4_addr_t end;
00491 };

```



```

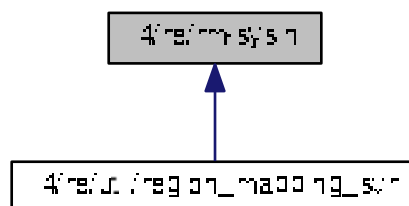
00577
00578 L4_RPC(long, get_regions, (l4_addr_t start,
00579 L4::Ipc::Ret_array<Region> regions));
00579 L4_RPC(long, get_areas, (l4_addr_t start,
00580 L4::Ipc::Ret_array<Area> areas));
00580
00581 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00582 L4::Cap<L4::Task> task, unsigned flags) const throw();
00583
00584 typedef L4::Typeid::Rpc<attach_t, detach_t, find_t,
00585 reserve_area_t, free_area_t,
00586 get_regions_t, get_areas_t> Rpc<
00587 >;
00588
00589
00590 inline int
00591 Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00592 L4::Cap<L4::Task> const &task) const throw()
00593 { return detach(addr, 1, mem, task, Detach_overlap); }
00594
00595 inline int
00596 Rm::detach(void *addr, L4::Cap<Dataspace> *mem,
00597 L4::Cap<L4::Task> const &task) const throw()
00598 { return detach((l4_addr_t)addr, 1, mem, task, Detach_overlap); }
00599
00600 inline int
00601 Rm::detach(l4_addr_t addr, unsigned long size,
00602 L4::Cap<Dataspace> *mem,
00603 L4::Cap<L4::Task> const &task) const throw()
00604 { return detach(addr, size, mem, task, Detach_exact); }
00605
00606

```

## 15.225 l4/re/rm-sys.h File Reference

Region mapper protocol definitions.

This graph shows which files directly or indirectly include this file:



### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Rm\\_::Opcodes](#)  
*Region-map communication-protocol opcodes.*

### 15.225.1 Detailed Description

Region mapper protocol definitions.

Definition in file [rm-sys.h](#).

## 15.226 rm-sys.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Rm_
00028 {
00034 enum Opcodes
00035 {
00036 Attach, Detach, Find, Attach_area, Detach_area, Get_regions, Get_areas
00037 };
00038 };
00039 };

```

## 15.227 l4/re/shared\_cap File Reference

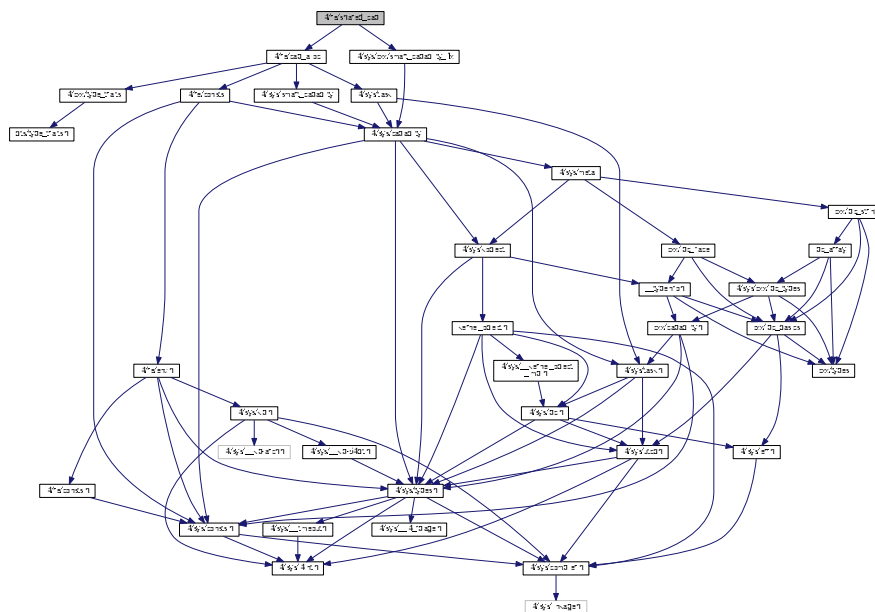
Shared\_cap / Shared\_del\_cap.

```

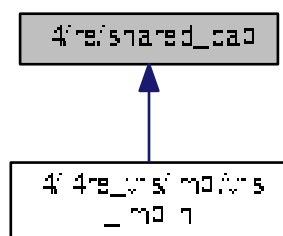
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_lx>

```

Include dependency graph for shared\_cap:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Typedefs

- `template<typename T>`  
using [L4Re::Shared\\_cap](#) = `L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4\_FP\_ALL\_SPACES > >`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
using [L4Re::shared\\_cap](#) = `L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4\_FP\_ALL\_SPACES > >`

*Shared capability that implements automatic free and unmap of the capability selector.*

- `template<typename T >`  
`using L4Re::Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ`  
`> >`

*Shared capability that implements automatic free and unmap+delete of the capability selector.*

- `template<typename T >`  
`using L4Re::shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE_OBJ`  
`> >`

*Shared capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T >`  
`Shared_cap< T > L4Re::make_shared_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in a Shared\_cap.*
- `template<typename T >`  
`Shared_del_cap< T > L4Re::make_shared_del_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in a Shared\_del\_cap.*

### 15.227.1 Detailed Description

Shared\_cap / Shared\_del\_cap.

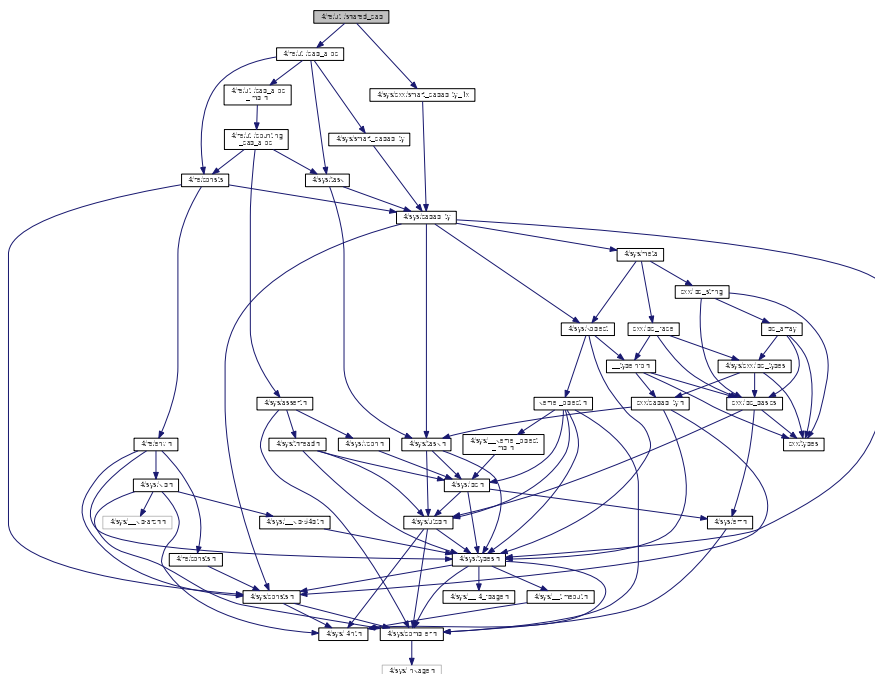
Definition in file [shared\\_cap](#).

## 15.228 shared\_cap

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2018 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re {
00029
00043 template< typename T >
00044 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>;
00046 template< typename T >
00047 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>;
00048
00058 template< typename T >
00059 Shared_cap<T>
00060 make_shared_cap(L4Re::Cap_alloc *ca)
00061 { return Shared_cap<T>(ca->alloc<T>(), ca); }
00062
00079 template< typename T >
00080 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>;
```

### 15.229 l4/re/util/shared\_cap File Reference

```
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
Include dependency graph for shared cap:
```



- **L4Re**  
*L4Re C++ Interfaces.*
- **L4Re::Util**

## Typedefs

- `template<typename T>`  
`using L4Re::Util::Shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES`  
`>>`

*Shared capability that implements automatic free and unmap of the capability selector.*

- `template<typename T>`  
`using L4Re::Util::shared_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_ALL_SPACES`  
`> >`

*Shared capability that implements automatic free and unmap of the capability selector.*

- `template<typename T>`  
`using L4Re::Util::Shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE↵`  
`E_OBJ > >`

*Shared capability that implements automatic free and unmap+delete of the capability selector.*

- `template<typename T>`  
`using L4Re::Util::shared_del_cap = L4::Detail::Shared_cap_impl< T, Smart_count_cap< L4_FP_DELETE↵`  
`E_OBJ > >`

*Shared capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Shared_cap< T> L4Re::Util::make_shared_cap ()`  
*Allocate a capability slot and wrap it in a Shared\_cap.*
- `template<typename T>`  
`Shared_del_cap< T> L4Re::Util::make_shared_del_cap ()`  
*Allocate a capability slot and wrap it in a Shared\_del\_cap.*

### 15.229.1 Detailed Description

Shared\_cap / Shared\_del\_cap.

Definition in file [shared\\_cap](#).

## 15.230 shared\_cap

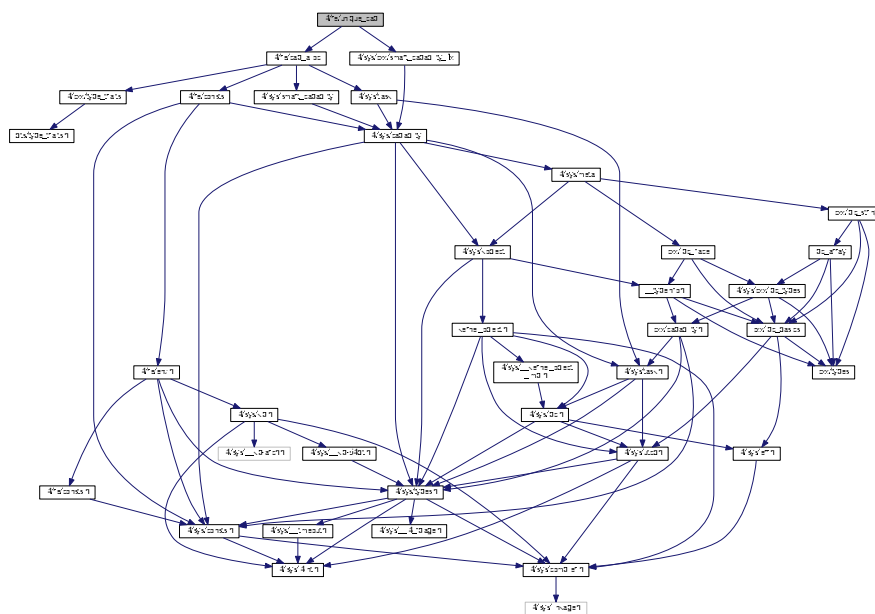
```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/re/util/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re { namespace Util {
00029
00058 template< typename T >
00059 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>;
00061 template< typename T >
00062 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>;
00063
```

```
00069 template< typename T >
00070 Shared_cap<T>
00071 make_shared_cap()
00072 { return Shared_cap<T>(cap_alloc.alloc<T>()); }
00073
00108 template< typename T >
00109 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>;
00111 template< typename T >
00112 using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>;
00113
00119 template< typename T >
00120 Shared_del_cap<T>
00121 make_shared_del_cap()
00122 { return Shared_del_cap<T>(cap_alloc.alloc<T>()); }
00123
00124 }} // namespace L4Re::Util
00125
```

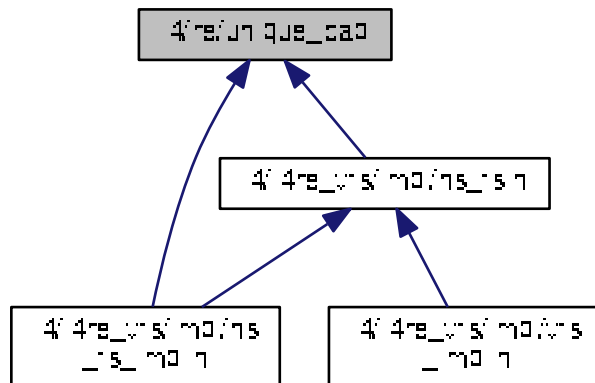
### 15.231 l4/re/unique\_cap File Reference

Unique\_cap / Unique\_del\_cap.

```
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
Include dependency graph for unique_cap:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Typedefs

- `template<typename T >`  
using [L4Re::Unique\\_cap](#) = `L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4\_FP\_ALL\_SPACES > >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
using [L4Re::unique\\_cap](#) = `L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4\_FP\_ALL\_SPACES > >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T >`  
using [L4Re::Unique\\_del\\_cap](#) = `L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4\_FP\_DELETE\_OBJ > >`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T >`  
using [L4Re::unique\\_del\\_cap](#) = `L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4\_FP\_DELETE\_OBJ > >`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T >`  
`Unique_cap< T > L4Re::make\_unique\_cap (L4Re::Cap\_alloc *ca)`  
*Allocate a capability slot and wrap it in an Unique\_cap.*
- `template<typename T >`  
`Unique_del_cap< T > L4Re::make\_unique\_del\_cap (L4Re::Cap\_alloc *ca)`  
*Allocate a capability slot and wrap it in an Unique\_del\_cap.*



### 15.231.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 15.232 unique\_cap

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/re/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_1x>
00027
00028 namespace L4Re {
00029
00041 template< typename T >
00042 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>;
00044 template< typename T >
00045 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>;
00046
00056 template< typename T >
00057 Unique_cap<T>
00058 make_unique_cap(L4Re::Cap_alloc *ca)
00059 { return Unique_cap<T>(ca->alloc<T>(), ca); }
00060
00074 template< typename T >
00075 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>;
00077 template<typename T>
00078 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>;
00079
00089 template< typename T >
00090 Unique_del_cap<T>
00091 make_unique_del_cap(L4Re::Cap_alloc *ca)
00092 { return Unique_del_cap<T>(ca->alloc<T>(), ca); }
00093
00094 }

```

## 15.233 l4/re/util/unique\_cap File Reference

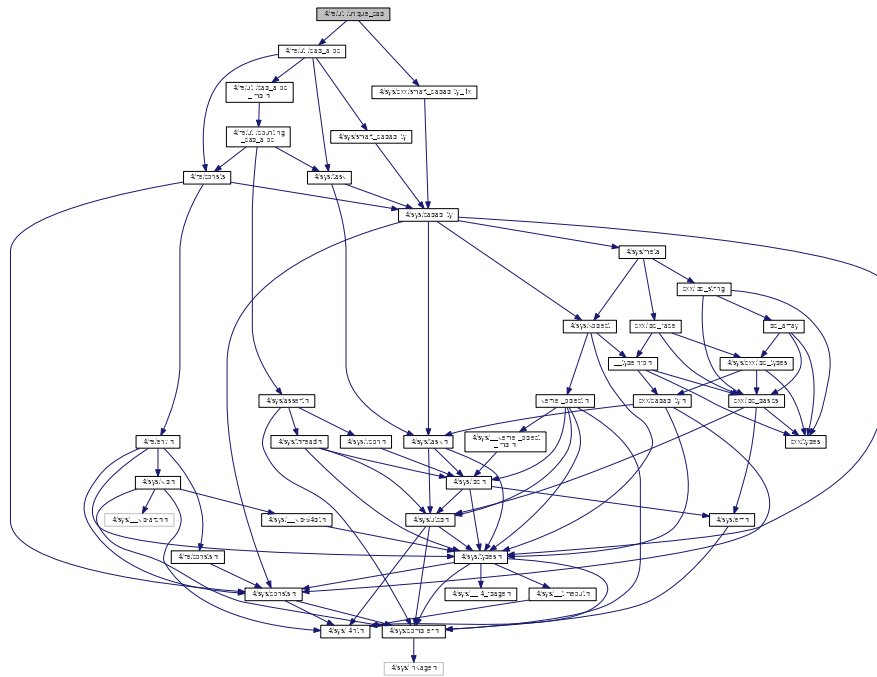
Unique\_cap / Unique\_del\_cap.

```

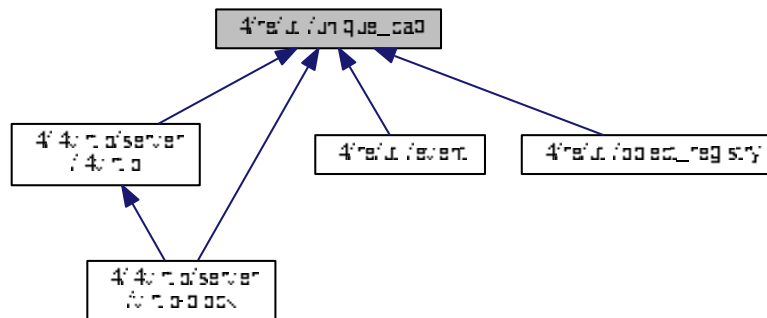
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>

```

Include dependency graph for `unique_cap`:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)

*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

## Typedefs

- `template<typename T>`  
`using L4Re::Util::Unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES`  
`> >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::Util::unique_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_ALL_SPACES`  
`> >`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::Util::Unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE↵`  
`_OBJ > >`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using L4Re::Util::unique_del_cap = L4::Detail::Unique_cap_impl< T, Smart_cap_auto< L4_FP_DELETE↵`  
`_OBJ > >`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Unique_cap< T > L4Re::Util::make_unique_cap ()`  
*Allocate a capability slot and wrap it in an Unique\_cap.*
- `template<typename T>`  
`Unique_del_cap< T > L4Re::Util::make_unique_del_cap ()`  
*Allocate a capability slot and wrap it in an Unique\_del\_cap.*

### 15.233.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 15.234 unique\_cap

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/re/util/cap_alloc>
00026 #include <l4/sys/cxx/smart_capability_lx>
```

```
00027
00028 namespace L4Re { namespace Util {
00029
00053 template< typename T >
00054 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>;
00056 template< typename T >
00057 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>;
00058
00064 template< typename T >
00065 Unique_cap<T>
00066 make_unique_cap()
00067 { return Unique_cap<T>(cap_alloc.alloc<T>()); }
00068
00096 template< typename T >
00097 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>;
00098 template< typename T >
00100 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>;
00101
00107 template< typename T >
00108 Unique_del_cap<T>
00109 make_unique_del_cap()
00110 { return Unique_del_cap<T>(cap_alloc.alloc<T>()); }
00111
00112 }}
00113
```

### 15.235 I4/re/util/bitmap\_cap\_alloc File Reference

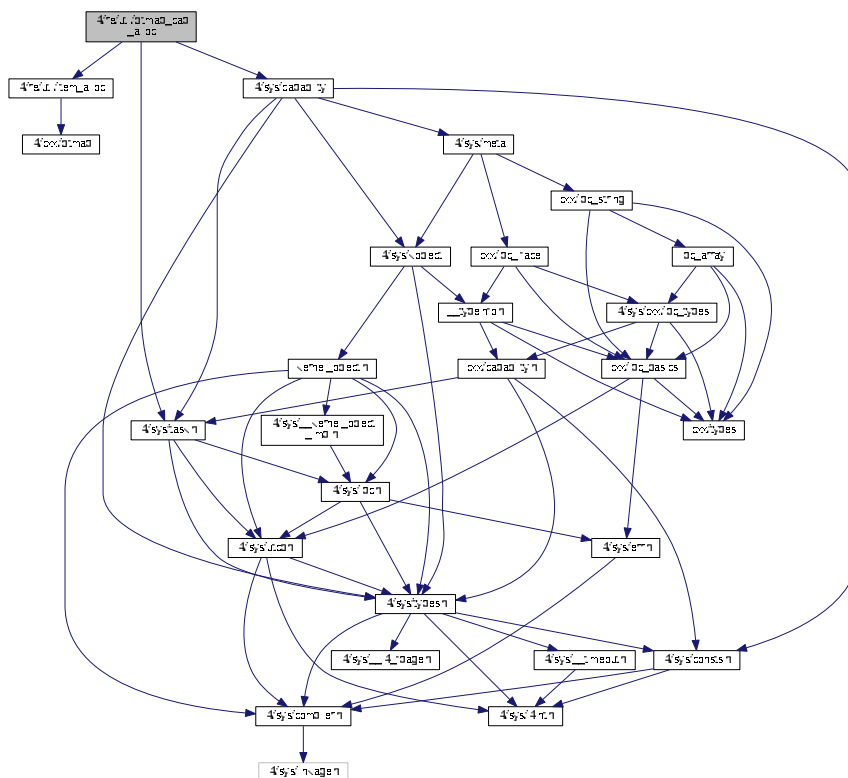
Bitmap capability allocator.

```
#include <14/re/util/item_alloc>
```

```
#include <linux/sys/capability>
```

```
#include <linux/sched/task.h>
```

Include dependency graph for bitmap\_cap\_alloc:



## Data Structures

- class [L4Re::Util::Cap\\_alloc\\_base](#)  
*Capability allocator.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

### 15.235.1 Detailed Description

Bitmap capability allocator.

Definition in file [bitmap\\_cap\\_alloc](#).

## 15.236 bitmap\_cap\_alloc

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/re/util/item_alloc>
00029 #include <l4/sys/capability>
00030 #include <l4/sys/task.h>
00031
00032 namespace L4Re { namespace Util {
00033
00038 class Cap_alloc_base
00039 {
00040 private:
00041 long _bias;
00042 Item_alloc_base _items;
00043
00044 public:
00045 enum State { Free = 0, Allocated, Unknown };
00046 Cap_alloc_base(long max, void *mem, long bias = 0)
00047 throw() : _bias(bias), _items(max, mem) {}
00048
00049 L4::Cap<void> alloc() throw()
00050 {
00051 long cap = _items.alloc();
00052 if (cap < 0)
00053 return L4::Cap<void>::Invalid;
00054 return L4::Cap<void>((cap + _bias) << L4_CAP_SHIFT);
00055 }
00056 }

```

```

00057
00058 long hint() const { return _items.hint(); }
00059
00063 template< typename T >
00064 L4::Cap<T> alloc() throw()
00065 { return L4::Cap<T>(alloc().cap()); }
00066
00067 State is_allocated(L4::Cap<void> c) const throw()
00068 {
00069 long idx = (c.cap() >> L4_CAP_SHIFT);
00070
00071 if (idx < _bias)
00072 return Unknown;
00073
00074 idx -= _bias;
00075 return _items.is_allocated(idx) ? Allocated : Free;
00076 }
00077
00081 template< typename T >
00082 void free(L4::Cap<T> const &cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00083 l4_umword_t unmap_flags = L4_FP_ALL_SPACES) throw()
00084 {
00085 long idx = (cap.cap() >> L4_CAP_SHIFT);
00086 if (idx < _bias)
00087 return;
00088
00089 idx -= _bias;
00090
00091 _items.free(idx);
00092
00093 if (l4_is_valid_cap(task))
00094 l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00095 }
00096
00097 // since we have no counters assume counter always > 0
00098 void take(L4::Cap<void>) throw() {}
00099 bool release(L4::Cap<void>, l4_cap_idx_t task =
L4_INVALID_CAP,
00100 unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
00101 { (void)task; (void)unmap_flags; return false; }
00102
00103 long last() throw()
00104 {
00105 return _items.size() + _bias - 1;
00106 }
00107 };
00108
00109 template< long Size >
00110 class Cap_alloc : public Cap_alloc_base
00111 {
00112 private:
00113 typename Bitmap_base::Word<Size>::Type _bits[Bitmap_base::Word<Size>::Size];
00114
00115 public:
00116 explicit Cap_alloc(long bias = 0) throw()
00117 : Cap_alloc_base(Size, _bits, bias) {}
00118
00119 };
00120
00121 }
00122 }

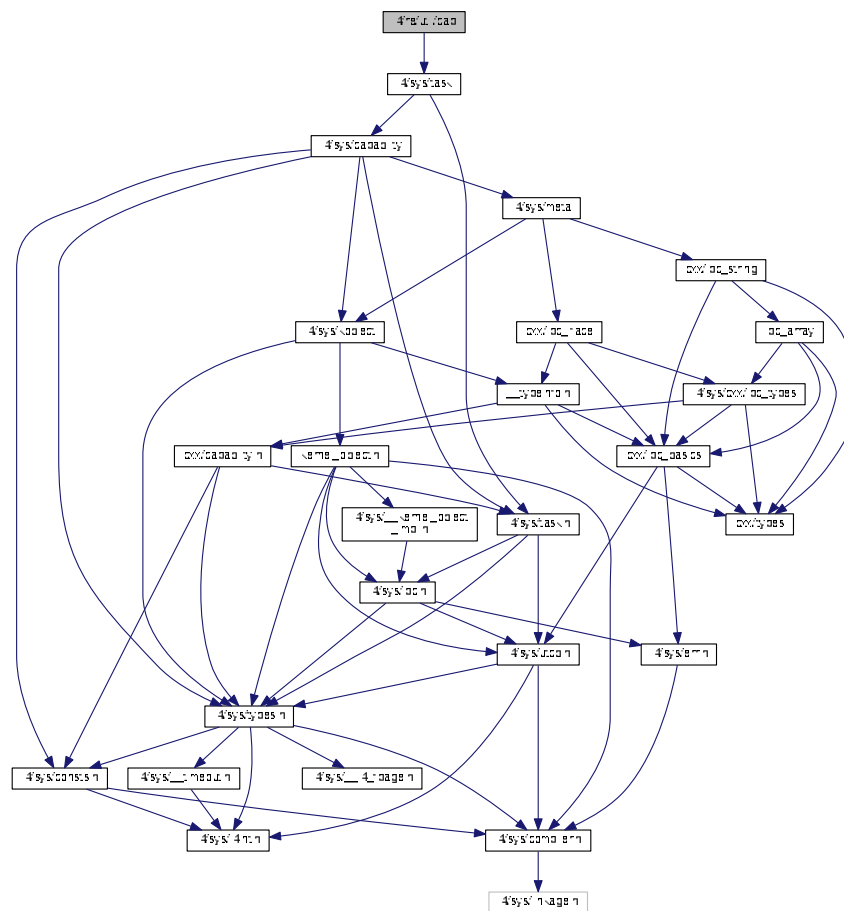
```

## 15.237 l4/re/util/cap File Reference

Capability utility functions.

```
#include <l4/sys/task>
```

Include dependency graph for cap:



## Namespaces

- **L4Re**  
*L4Re C++ Interfaces.*
- **L4Re::Util**  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

### 15.237.1 Detailed Description

### Capability utility functions.

Definition in file [cap.](#)

**15.238 cap**

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028
00029 namespace L4Re { namespace Util {
00030
00031 L4_CV static inline l4_msgtag_t cap_release(L4::Cap<void> cap)
00032 {
00033 return l4_task_unmap(L4_BASE_TASK_CAP,
00034 l4_obj_fpage(cap.cap(), 0, L4_FPAGE_RWX),
00035 L4_FP_ALL_SPACES);
00036 }
00037
00038 }
00039
00040 }

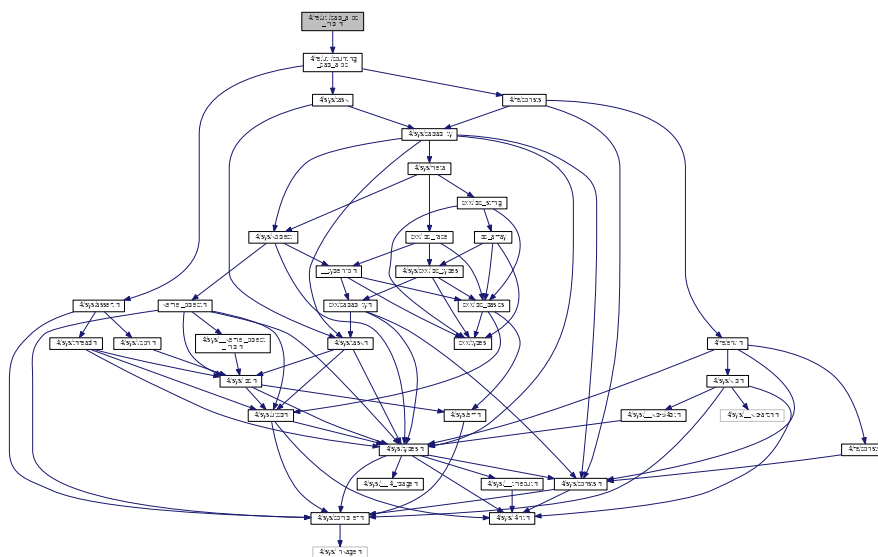
```

### 15.239 l4/re/util/cap\_alloc\_impl.h File Reference

### Capability allocator implementation.

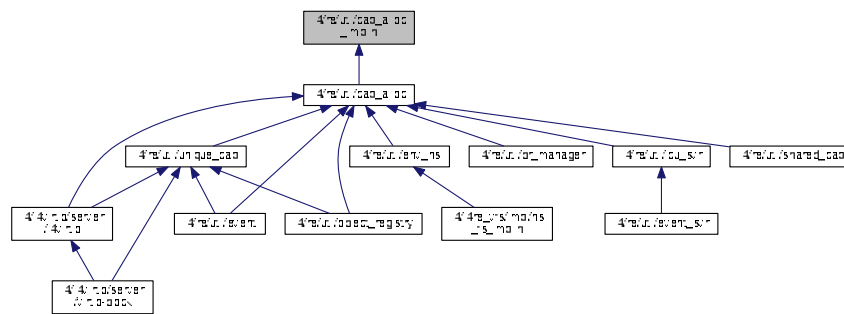
```
#include <linux/re/utl/counting_cap_alloc>
```

Include dependency graph for cap\_alloc\_impl.h:





This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)

*Documentation of the [L4 Runtime Environment](#) utility functionality in C++.*

### 15.239.1 Detailed Description

Capability allocator implementation.

Definition in file [cap\\_alloc\\_impl.h](#).

## 15.240 cap\_alloc\_impl.h

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021
00022 #define L4RE_STATIC_CAP_ALLOC
00023 #if defined(L4RE_STATIC_CAP_ALLOC)
00024
00025 #include <l4/re/util/bitmap_cap_alloc>
00026
00027 namespace L4Re { namespace Util {
00028
00029 typedef Cap_alloc_base _Cap_alloc;
00030
00031

```

```

00036 }}
00037
00038 #else
00039 #include <l4/re/util/counting_cap_alloc>
00040
00041 namespace L4Re { namespace Util {
00042
00043 typedef Counting_cap_alloc<L4Re::Util::Counter<unsigned char> > _Cap_alloc;
00044
00045 }}
00046 #endif
00047
00048

```

## 15.241 l4/re/util/counting\_cap\_alloc File Reference

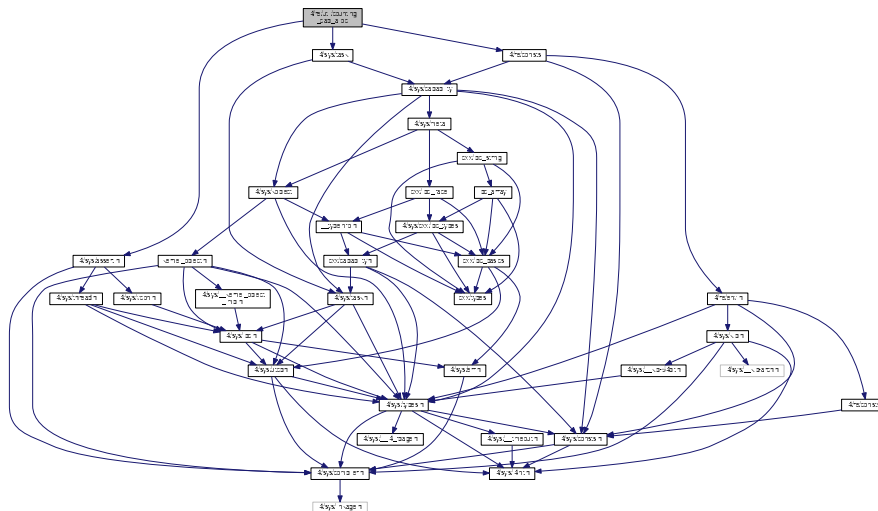
Reference-counting capability allocator.

```

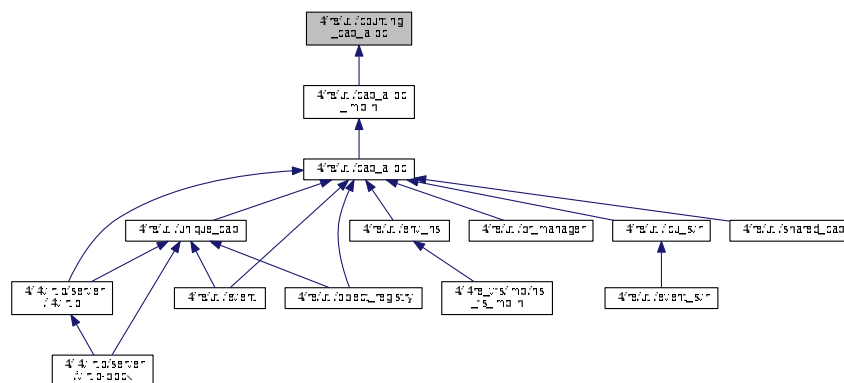
#include <l4/sys/task>
#include <l4/sys/assert.h>
#include <l4/re/consts>

```

Include dependency graph for counting\_cap\_alloc:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4Re::Util::Counter< COUNTER >](#)  
*Counter for Counting\_cap\_alloc with variable data width.*
- class [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >](#)  
*Internal reference-counting cap allocator.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

### 15.241.1 Detailed Description

Reference-counting capability allocator.

Definition in file [counting\\_cap\\_alloc](#).

## 15.242 counting\_cap\_alloc

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023
00024 #pragma once
00025
00026 #include <l4/sys/task>
00027 #include <l4/sys/assert.h>
00028 #include <l4/re/consts>
00029
00030 namespace L4Re { namespace Util {
00031
00032 template< typename COUNTER = unsigned char >
00033 struct Counter
00034 {
00035 typedef COUNTER Type;
00036 Type _cnt;
00037
00038 static Type nil() { return 0; }
00039
00040 void free() { _cnt = 0; }
00041 bool is_free() const { return _cnt == 0; }
00042 void inc() { ++_cnt; }
00043 Type dec() { return --_cnt; }
00044 void alloc() { _cnt = 1; }
00045 };
00046
00047 template <typename COUNTERTYPE = L4Re::Util::Counter<unsigned char> >
00048 class Counting_cap_alloc

```

```

00076 {
00077 private:
00078 void operator = (Counting_cap_alloc const &) { }
00079 typedef COUNTERTYPE Counter;
00080
00081 COUNTERTYPE *_items;
00082 long _free_hint;
00083 long _bias;
00084 long _capacity;
00085
00086
00087 public:
00088
00089 template <unsigned COUNT>
00090 struct Counter_storage
00091 {
00092 COUNTERTYPE _buf[COUNT];
00093 typedef COUNTERTYPE Buf_type[COUNT];
00094 enum { Size = COUNT };
00095 };
00096
00097 protected:
00098
00104 Counting_cap_alloc() throw()
00105 : _items(0), _free_hint(0), _bias(0), _capacity(0)
00106 {}
00107
00121 void setup(void *m, long capacity, long bias) throw()
00122 {
00123 _items = (Counter*)m;
00124 _capacity = capacity;
00125 _bias = bias;
00126 }
00127
00128 public:
00135 L4::Cap<void> alloc() throw()
00136 {
00137 if (_free_hint >= _capacity)
00138 return L4::Cap_base::Invalid;
00139
00140 for (long i = _free_hint; i < _capacity; ++i)
00141 {
00142 if (_items[i].is_free())
00143 {
00144 _items[i].alloc();
00145 _free_hint = i + 1;
00146
00147 return L4::Cap<void>((i + _bias) << L4_CAP_SHIFT);
00148 }
00149 }
00150
00151 return L4::Cap<void>::Invalid;
00152 }
00153
00155 template <typename T>
00156 L4::Cap<T> alloc() throw()
00157 {
00158 return L4::cap_cast<T>(alloc());
00159 }
00160
00161
00171 void take(L4::Cap<void> cap) throw()
00172 {
00173 long c = cap.cap() >> L4_CAP_SHIFT;
00174 if (c < _bias)
00175 return;
00176
00177 c -= _bias;
00178 if (c >= _capacity)
00179 return;
00180
00181 _items[c].inc();
00182 }
00183
00184
00198 bool free(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00199 unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
00200 {
00201 long c = cap.cap() >> L4_CAP_SHIFT;
00202 if (c < _bias)
00203 return false;
00204
00205 c -= _bias;
00206
00207 if (c >= _capacity)
00208 return false;

```

```

00209
00210 l4_assert(!_items[c].is_free());
00211
00212 if (l4_is_valid_cap(task))
00213 l4_task_unmap(task, cap.fpage(), unmap_flags);
00214
00215 if (c < _free_hint)
00216 _free_hint = c;
00217
00218 _items[c].free();
00219
00220 return true;
00221 }
00222
00241 bool release(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00242 unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
00243 {
00244 long c = cap.cap() >> L4_CAP_SHIFT;
00245 if (c < _bias)
00246 return false;
00247
00248 c -= _bias;
00249
00250 if (c >= _capacity)
00251 return false;
00252
00253 l4_assert(!_items[c].is_free());
00254
00255 if (_items[c].dec() == Counter::nil())
00256 {
00257 if (task != L4_INVALID_CAP)
00258 l4_task_unmap(task, cap.fpage(), unmap_flags);
00259
00260 if (c < _free_hint)
00261 _free_hint = c;
00262
00263 return true;
00264 }
00265 return false;
00266 }
00267
00268
00272 long last() throw()
00273 {
00274 return _capacity + _bias - 1;
00275 }
00276 };
00277
00278 }}
00279

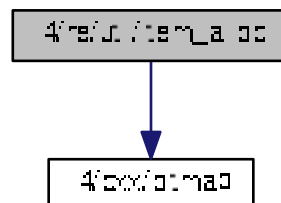
```

## 15.243 l4/re/util/item\_alloc File Reference

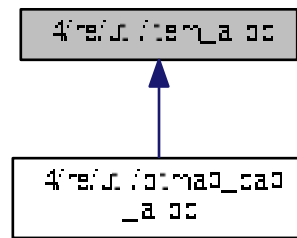
Item allocator.

```
#include <l4/cxx/bitmap>
```

Include dependency graph for item\_alloc:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Util::Item\\_alloc\\_base](#)  
*Item allocator.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

### 15.243.1 Detailed Description

Item allocator.

Definition in file [item\\_alloc](#).

## 15.244 item\_alloc

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005 * Alexander Warg <warg@os.inf.tu-dresden.de>
00006 * economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */

```

```

00024 */
00025
00026 #pragma once
00027
00028 #include <l4/cxx/bitmap>
00029
00030 namespace L4Re { namespace Util {
00031
00032 using cxx::Bitmap_base;
00033 using cxx::Bitmap;
00034
00038 class Item_alloc_base
00039 {
00040 private:
00041 long _capacity;
00042 long _free_hint;
00043 Bitmap_base _bits;
00044
00045 public:
00046 bool is_allocated(long item) const throw()
00047 { return _bits[item]; }
00048
00049 long hint() const { return _free_hint; }
00050
00051 bool alloc(long item) throw()
00052 {
00053 if (!_bits[item])
00054 {
00055 _bits.set_bit(item);
00056 return true;
00057 }
00058 return false;
00059 }
00060
00061 void free(long item) throw()
00062 {
00063 if (item < _free_hint)
00064 _free_hint = item;
00065
00066 _bits.clear_bit(item);
00067 }
00068
00069 Item_alloc_base(long size, void *mem) throw()
00070 : _capacity(size), _free_hint(0), _bits(mem)
00071 {}
00072
00073 long alloc() throw()
00074 {
00075 if (_free_hint >= _capacity)
00076 return -1;
00077
00078 long free = _bits.scan_zero(_capacity, _free_hint);
00079 if (free >= 0)
00080 {
00081 _bits.set_bit(free);
00082 _free_hint += 1;
00083 }
00084 return free;
00085 }
00086
00087 long size() const throw()
00088 {
00089 return _capacity;
00090 }
00091 };
00092
00093 template< long Bits >
00094 class Item_alloc : public Item_alloc_base
00095 {
00096 private:
00097 typename Bitmap_base::Word<Bits>::Type _bits[Bitmap_base::Word<Bits>::Size];
00098
00099 public:
00100 Item_alloc() throw() : Item_alloc_base(Bits, _bits) {}
00101 };
00102
00103 }

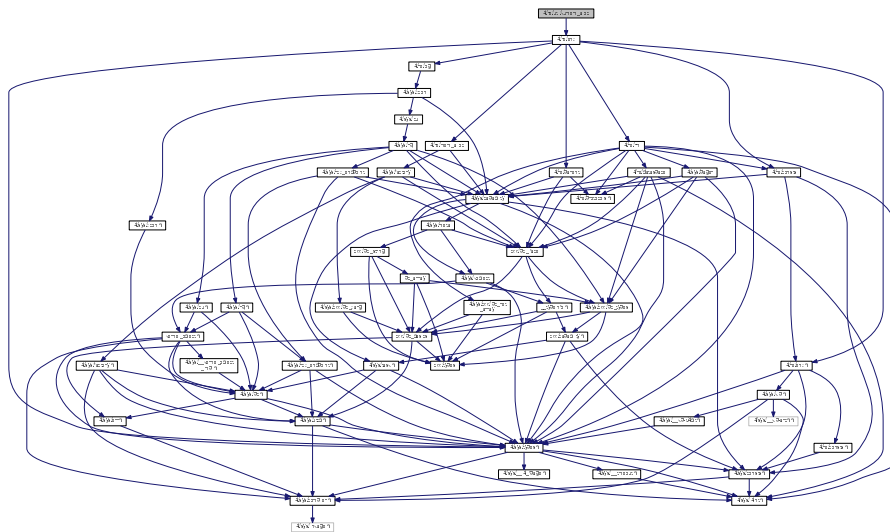
```

## 15.245 I4/re/util/kumem\_alloc File Reference

Kumem allocator helper.

```
#include <l4/re/env>
```

Include dependency graph for kumem\_alloc:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*
- [L4Re::Util](#)

*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

## Functions

- `int L4Re::Util::kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) throw ()`  
*Allocate state area.*

### 15.245.1 Detailed Description

Kumem allocator helper.

Definition in file [kumem\\_alloc](#).

## 15.246 kumem\_alloc

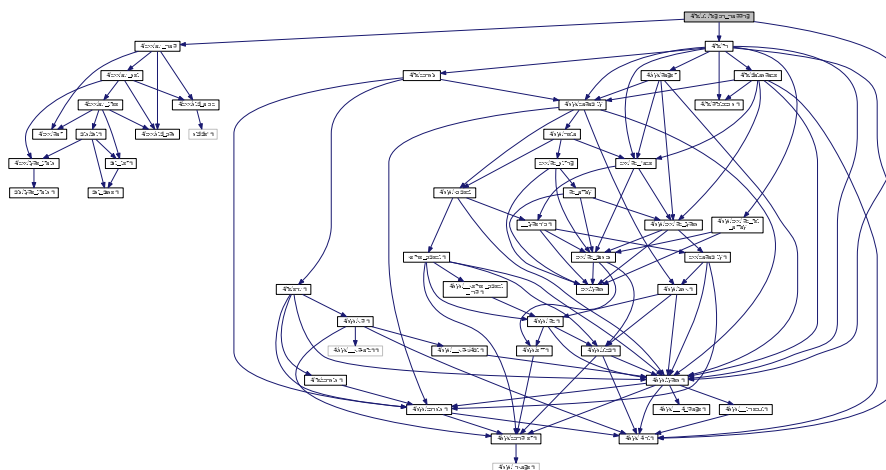
```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005 * Alexander Warg <warg@os.inf.tu-dresden.de>
00006 * economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
```



```
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/re/env>
00029
00030 namespace L4Re { namespace Util {
00031
00037
00055 int
00056 kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00057 L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00058 L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) throw();
00059
00061 }}
```

### Region handling.

```
#include <l4/cxx/avl_map>
#include <l4/sys/types.h>
#include <l4/re/rm>
Include dependency graph for region mapping:
```





```

00035 class Region
00036 {
00037 private:
00038 l4_addr_t _start, _end;
00039
00040 public:
00041 Region() throw() : _start(~0UL), _end(~0UL) {}
00042 Region(l4_addr_t addr) throw() : _start(addr), _end(addr) {}
00043 Region(l4_addr_t start, l4_addr_t end) throw()
00044 : _start(start), _end(end) {}
00045 l4_addr_t start() const throw() { return _start; }
00046 l4_addr_t end() const throw() { return _end; }
00047 unsigned long size() const throw() { return end() - start() + 1; }
00048 bool invalid() const throw() { return _start == ~0UL && _end == ~0UL; }
00049 bool operator < (Region const &o) const throw()
00050 { return end() < o.start(); }
00051 bool contains(Region const &o) const throw()
00052 { return o.start() >= start() && o.end() <= end(); }
00053 bool operator == (Region const &o) const throw()
00054 { return o.start() == start() && o.end() == end(); }
00055 ~Region() throw() {}
00056 };
00057
00058 template< typename DS, typename OPS >
00059 class Region_handler
00060 {
00061 private:
00062 l4_addr_t _offs;
00063 DS _mem;
00064 l4_cap_idx_t _client_cap = L4_INVALID_CAP;
00065 unsigned short _flags;
00066
00067 public:
00068 typedef DS Dataspace;
00069 typedef OPS Ops;
00070 typedef typename OPS::Map_result Map_result;
00071
00072 Region_handler() throw() : _offs(0), _mem(), _flags() {}
00073 Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
00074 l4_addr_t offset = 0, unsigned flags = 0) throw()
00075 : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
00076 {}
00077 Dataspace const &memory() const throw() { return _mem; }
00078 l4_cap_idx_t client_cap_idx() const throw() { return _client_cap; }
00079 l4_addr_t offset() const throw() { return _offs; }
00080 l4_addr_t is_ro() const throw() { return _flags & L4Re::Rm::Read_only; }
00081 unsigned long caching() const throw() { return _flags & L4Re::Rm::Caching; }
00082 unsigned flags() const throw() { return _flags; }
00083
00084 Region_handler operator + (long offset) const throw()
00085 { Region_handler n = *this; n._offs += offset; return n; }
00086
00087 void unmap(l4_addr_t va, l4_addr_t ds_offs, unsigned long size) const throw()
00088 { Ops::unmap(this, va, ds_offs, size); }
00089
00090 void free(l4_addr_t start, unsigned long size) const throw()
00091 { Ops::free(this, start, size); }
00092
00093 void take() const { Ops::take(this); }
00094 void release() const { Ops::release(this); }
00095
00096 int map(l4_addr_t addr, Region const &r, bool writable, Map_result *result) const
00097 { return Ops::map(this, addr, r, writable, result); }
00098
00099 };
00100
00101
00102 template< typename Hdlr, template<typename T> class Alloc >
00103 class Region_map
00104 {
00105 protected:
00106 typedef cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >
00107 Tree;
00108 Tree _rm;
00109 Tree _am;
00110 private:
00111 l4_addr_t _start;
00112 l4_addr_t _end;
00113
00114 protected:
00115 void set_limits(l4_addr_t start, l4_addr_t end) throw()
00116 {
00117 _start = start;
00118 _end = end;
00119 }
00120

```

```

00121 public:
00122 typedef typename Tree::Item_type Item;
00123 typedef typename Tree::Node Node;
00124 typedef typename Tree::Key_type Key_type;
00125 typedef Hdlr Region_handler;
00126
00127 typedef typename Tree::Iterator Iterator;
00128 typedef typename Tree::Const_iterator Const_iterator;
00129 typedef typename Tree::Rev_iterator Rev_iterator;
00130 typedef typename Tree::Const_rev_iterator Const_rev_iterator;
00131
00132 Iterator begin() throw() { return _rm.begin(); }
00133 Const_iterator begin() const throw() { return _rm.begin(); }
00134 Iterator end() throw() { return _rm.end(); }
00135 Const_iterator end() const throw() { return _rm.end(); }
00136
00137 Iterator area_begin() throw() { return _am.begin(); }
00138 Const_iterator area_begin() const throw() { return _am.begin(); }
00139 Iterator area_end() throw() { return _am.end(); }
00140 Const_iterator area_end() const throw() { return _am.end(); }
00141 Node area_find(Key_type const &c) const throw() { return _am.find_node(c); }
00142
00143 enum Attach_flags
00144 {
00145 None = 0,
00146 Search = L4Re::Rm::Search_addr,
00147 In_area = L4Re::Rm::In_area,
00148 };
00149
00150 l4_addr_t min_addr() const throw() { return _start; }
00151 l4_addr_t max_addr() const throw() { return _end; }
00152
00153
00154 Region_map(l4_addr_t start, l4_addr_t end) throw() : _start(start), _end(end) {}
00155
00156 Node find(Key_type const &key) const throw()
00157 {
00158 Node n = _rm.find_node(key);
00159 if (!n)
00160 return Node();
00161
00162 // 'find' should find any region overlapping with the searched one, the
00163 // caller should check for further requirements
00164 if (0)
00165 if (!n->first.contains(key))
00166 return Node();
00167
00168 return n;
00169 }
00170
00171 Node lower_bound(Key_type const &key) const throw()
00172 {
00173 Node n = _rm.lower_bound_node(key);
00174 return n;
00175 }
00176
00177 Node lower_bound_area(Key_type const &key) const throw()
00178 {
00179 Node n = _am.lower_bound_node(key);
00180 return n;
00181 }
00182
00183 l4_addr_t attach_area(l4_addr_t addr, unsigned long size,
00184 unsigned flags = None,
00185 unsigned char align = L4_PAGESHIFT) throw()
00186 {
00187 if (size < 2)
00188 return L4_INVALID_ADDR;
00189
00190 Region c;
00191
00192 if (!(flags & Search))
00193 {
00194 c = Region(addr, addr + size - 1);
00195 Node r = _am.find_node(c);
00196 if (r)
00197 return L4_INVALID_ADDR;
00198 }
00199
00200 while (flags & Search)
00201 {
00202 if (addr < min_addr() || (addr + size - 1) > max_addr())
00203 addr = min_addr();
00204 addr = find_free(addr, max_addr(), size, align, flags);
00205 if (addr == L4_INVALID_ADDR)
00206 return L4_INVALID_ADDR;
00207 }

```

```

00208
00209 c = Region(addr, addr + size - 1);
00210 Node r = _am.find_node(c);
00211 if (!r)
00212 break;
00213
00214 if (r->first.end() >= max_addr())
00215 return L4_INVALID_ADDR;
00216
00217 addr = r->first.end() + 1;
00218 }
00219
00220 if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, flags)).second == 0)
00221 return addr;
00222
00223 return L4_INVALID_ADDR;
00224 }
00225
00226 bool detach_area(l4_addr_t addr) throw()
00227 {
00228 if (_am.remove(addr))
00229 return false;
00230
00231 return true;
00232 }
00233
00234 void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
00235 unsigned flags = None, unsigned char align = L4_PAGESHIFT) throw()
00236 {
00237 if (size < 2)
00238 return L4_INVALID_PTR;
00239
00240 l4_addr_t end = max_addr();
00241 l4_addr_t beg = (l4_addr_t)addr;
00242
00243 if (flags & In_area)
00244 {
00245 Node r = _am.find_node(Region(beg, beg + size - 1));
00246 if (!r || (r->second.flags() & L4Re::Rm::Reserved))
00247 return L4_INVALID_PTR;
00248
00249 end = r->first.end();
00250 }
00251
00252 if (flags & Search)
00253 {
00254 beg = find_free(beg, end, size, align, flags);
00255 if (beg == L4_INVALID_ADDR)
00256 return L4_INVALID_PTR;
00257 }
00258
00259 if (!(flags & (Search | In_area)) && _am.find_node(Region(beg, beg + size - 1)))
00260 return L4_INVALID_PTR;
00261
00262 if (beg < min_addr() || beg + size - 1 > end)
00263 return L4_INVALID_PTR;
00264
00265 if (_rm.insert(Region(beg, beg + size - 1), hdlr).second == 0)
00266 return (void*)beg;
00267
00268 return L4_INVALID_PTR;
00269 }
00270
00271 int detach(void *addr, unsigned long sz, unsigned flags,
00272 Region *reg, Hdlr *hdlr) throw()
00273 {
00274 Region dr((l4_addr_t)addr, (l4_addr_t)addr + sz - 1);
00275 Region res(~0UL, 0);
00276
00277 Node r = find(dr);
00278 if (!r)
00279 return -L4_ENOENT;
00280
00281 Region g = r->first;
00282 Hdlr const &h = r->second;
00283
00284 if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
00285 {
00286 if (_rm.remove(g))
00287 return -L4_ENOENT;
00288
00289 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() &
L4Re::Rm::Detach_free))
00290 h.free(0, g.size());
00291
00292 if (hdlr) *hdlr = h;
00293 if (reg) *reg = g;

```

```

00294
00295 if (find(dr))
00296 return Rm::Detached_ds | Rm::Detach_again;
00297 else
00298 return Rm::Detached_ds;
00299 }
00300 else if (dr.start() <= g.start())
00301 {
00302 // move the start of a region
00303
00304 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00305 h.free(0, dr.end() + 1 - g.start());
00306
00307 unsigned long sz = dr.end() + 1 - g.start();
00308 Item *cn = const_cast<Item*>((Item const *)r);
00309 cn->first = Region(dr.end() + 1, g.end());
00310 cn->second = cn->second + sz;
00311 if (hdlr) *hdlr = Hdlr();
00312 if (reg) *reg = Region(g.start(), dr.end());
00313 if (find(dr))
00314 return Rm::Kept_ds | Rm::Detach_again;
00315 else
00316 return Rm::Kept_ds;
00317 }
00318 else if (dr.end() >= g.end())
00319 {
00320 // move the end of a region
00321
00322 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00323 h.free(dr.start() - g.start(), g.end() + 1 - dr.start());
00324
00325 Item *cn = const_cast<Item*>((Item const *)r);
00326 cn->first = Region(g.start(), dr.start() - 1);
00327 if (hdlr) *hdlr = Hdlr();
00328 if (reg) *reg = Region(dr.start(), g.end());
00329
00330 if (find(dr))
00331 return Rm::Kept_ds | Rm::Detach_again;
00332 else
00333 return Rm::Kept_ds;
00334 }
00335 else if (g.contains(dr))
00336 {
00337 // split a single region that contains the new region
00338
00339 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00340 h.free(dr.start() - g.start(), dr.size());
00341
00342 // first move the end off the existing region before the new one
00343 const_cast<Item*>((Item const *)r)->first = Region(g.start(), dr.start() - 1);
00344
00345 int err;
00346
00347 // insert a second region for the remaining tail of
00348 // the old existing region
00349 err = _rm.insert(Region(dr.end() + 1, g.end()), h + (dr.end() + 1 - g.start())).second;
00350
00351 if (err)
00352 return err;
00353
00354 if (hdlr) *hdlr = h;
00355 if (reg) *reg = dr;
00356 return Rm::Split_ds;
00357 }
00358 return -L4_ENOENT;
00359 }
00360
00361 l4_addr_t find_free(l4_addr_t start, l4_addr_t end,
00362 l4_addr_t size,
00363 unsigned char align, unsigned flags) const throw();
00364 };
00365
00366
00367 template< typename Hdlr, template<typename T> class Alloc >
00368 l4_addr_t
00369 Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
00370 unsigned long size, unsigned char align, unsigned flags) const throw()
00371 {
00372 l4_addr_t addr = start;
00373
00374 if (addr == ~0UL || addr < min_addr() || addr >= end)
00375 addr = min_addr();
00376
00377 addr = l4_round_size(addr, align);
00378 Node r;
00379

```

```

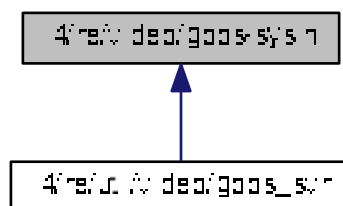
00380 for(;;)
00381 {
00382 if (addr > 0 && addr - 1 > end - size)
00383 return L4_INVALID_ADDR;
00384
00385 Region c(addr, addr + size - 1);
00386 r = _rm.find_node(c);
00387
00388 if (!r)
00389 {
00390 if (!(flags & In_area) && (r = _am.find_node(c)))
00391 {
00392 if (r->first.end() > end - size)
00393 return L4_INVALID_ADDR;
00394
00395 addr = l4_round_size(r->first.end() + 1, align);
00396 continue;
00397 }
00398 break;
00399 }
00400 else if (r->first.end() > end - size)
00401 return L4_INVALID_ADDR;
00402
00403 addr = l4_round_size(r->first.end() + 1, align);
00404 }
00405
00406 if (!r)
00407 return addr;
00408
00409 return L4_INVALID_ADDR;
00410 }
00411
00412 }}

```

## 15.249 l4/re/video/goos-sys.h File Reference

Goos protocol definition.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Video::Goos\\_::Opcodes](#)  
*Frame buffer communication-protocol opcodes.*

### 15.249.1 Detailed Description

Goos protocol definition.

Definition in file [goos-sys.h](#).

## 15.250 goos-sys.h

```

00001 #pragma once
00002
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 namespace L4Re { namespace Video {
00027 namespace Goos_
00028 {
00034 enum Opcodes
00035 {
00036 Info, Get_buffer, Create_buffer, Create_view,
00037 Delete_buffer, Delete_view,
00038 View_info, View_set_info, View_stack, View_refresh,
00039 Screen_refresh
00040 };
00041 };
00042 }}
```

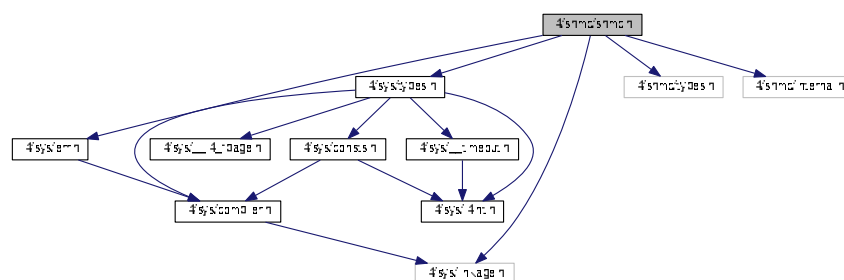
## 15.251 l4/shmc/shmc.h File Reference

Shared memory library header file.

```

#include <l4/sys/linkage.h>
#include <l4/sys/types.h>
#include <l4/sys/err.h>
#include <l4/shmc/types.h>
#include <l4/shmc/internal.h>
```

Include dependency graph for shmc.h:





## Functions

- long [l4shmc\\_create](#) (const char \*shmc\_name, [l4\\_umword\\_t](#) shm\_size)  
*Create a shared memory area.*
- long [l4shmc\\_attach](#) (const char \*shmc\_name, [l4shmc\\_area\\_t](#) \*shmarea)  
*Attach to a shared memory area.*
- long [l4shmc\\_attach\\_to](#) (const char \*shmc\_name, [l4\\_umword\\_t](#) timeout\_ms, [l4shmc\\_area\\_t](#) \*shmarea)  
*Attach to a shared memory area, with limited waiting.*
- long [l4shmc\\_add\\_chunk](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*chunk\_name, [l4\\_umword\\_t](#) chunk\_capacity, [l4shmc\\_chunk\\_t](#) \*chunk)  
*Add a chunk in the shared memory area.*
- long [l4shmc\\_add\\_signal](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*signal\_name, [l4shmc\\_signal\\_t](#) \*signal)  
*Add a signal for the shared memory area.*
- long [l4shmc\\_trigger](#) ([l4shmc\\_signal\\_t](#) \*signal)  
*Trigger a signal.*
- long [l4shmc\\_chunk\\_try\\_to\\_take](#) ([l4shmc\\_chunk\\_t](#) \*chunk)  
*Try to mark chunk busy.*
- long [l4shmc\\_chunk\\_ready](#) ([l4shmc\\_chunk\\_t](#) \*chunk, [l4\\_umword\\_t](#) size)  
*Mark chunk as filled (ready).*
- long [l4shmc\\_chunk\\_ready\\_sig](#) ([l4shmc\\_chunk\\_t](#) \*chunk, [l4\\_umword\\_t](#) size)  
*Mark chunk as filled (ready) and signal consumer.*
- long [l4shmc\\_get\\_chunk](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*chunk\_name, [l4shmc\\_chunk\\_t](#) \*chunk)  
*Get chunk out of shared memory area.*
- long [l4shmc\\_get\\_chunk\\_to](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*chunk\_name, [l4\\_umword\\_t](#) timeout\_ms, [l4shmc\\_chunk\\_t](#) \*chunk)  
*Get chunk out of shared memory area, with timeout.*
- long [l4shmc\\_iterate\\_chunk](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*\*chunk\_name, long offs)  
*Iterate over names of all existing chunks.*
- long [l4shmc\\_attach\\_signal](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*signal\_name, [l4\\_cap\\_idx\\_t](#) thread, [l4shmc\\_signal\\_t](#) \*signal)  
*Attach to signal.*
- long [l4shmc\\_attach\\_signal\\_to](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*signal\_name, [l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) timeout\_ms, [l4shmc\\_signal\\_t](#) \*signal)  
*Attach to signal, with timeout.*
- long [l4shmc\\_get\\_signal\\_to](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*signal\_name, [l4\\_umword\\_t](#) timeout\_ms, [l4shmc\\_signal\\_t](#) \*signal)  
*Get signal object from the shared memory area.*
- long [l4shmc\\_enable\\_signal](#) ([l4shmc\\_signal\\_t](#) \*signal)  
*Enable a signal.*
- long [l4shmc\\_enable\\_chunk](#) ([l4shmc\\_chunk\\_t](#) \*chunk)  
*Enable a signal connected with a chunk.*
- long [l4shmc\\_wait\\_any](#) ([l4shmc\\_signal\\_t](#) \*\*retsignal)  
*Wait on any signal.*
- long [l4shmc\\_wait\\_any\\_try](#) ([l4shmc\\_signal\\_t](#) \*\*retsignal)  
*Check whether any waited signal has an event pending.*
- long [l4shmc\\_wait\\_any\\_to](#) ([l4\\_timeout\\_t](#) timeout, [l4shmc\\_signal\\_t](#) \*\*retsignal)  
*Wait for any signal with timeout.*
- long [l4shmc\\_wait\\_signal](#) ([l4shmc\\_signal\\_t](#) \*signal)  
*Wait on a specific signal.*
- long [l4shmc\\_wait\\_signal\\_to](#) ([l4shmc\\_signal\\_t](#) \*signal, [l4\\_timeout\\_t](#) timeout)  
*Wait on a specific signal, with timeout.*
- long [l4shmc\\_wait\\_signal\\_try](#) ([l4shmc\\_signal\\_t](#) \*signal)

- Check whether a specific signal has an event pending.*

  - long [l4shmc\\_wait\\_chunk](#) (l4shmc\_chunk\_t \*chunk)

*Wait on a specific chunk.*

  - long [l4shmc\\_wait\\_chunk\\_to](#) (l4shmc\_chunk\_t \*chunk, [l4\\_timeout\\_t](#) timeout)

*Check whether a specific chunk has an event pending, with timeout.*

  - long [l4shmc\\_wait\\_chunk\\_try](#) (l4shmc\_chunk\_t \*chunk)

*Check whether a specific chunk has an event pending.*

  - long [l4shmc\\_chunk\\_consumed](#) (l4shmc\_chunk\_t \*chunk)

*Mark a chunk as free.*

  - long [l4shmc\\_connect\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk, l4shmc\_signal\_t \*signal)

*Connect a signal with a chunk.*

  - long [l4shmc\\_is\\_chunk\\_ready](#) (l4shmc\_chunk\_t \*chunk)

*Check whether data is available.*

  - long [l4shmc\\_is\\_chunk\\_clear](#) (l4shmc\_chunk\_t \*chunk)

*Check whether chunk is free.*

  - void \* [l4shmc\\_chunk\\_ptr](#) (l4shmc\_chunk\_t \*chunk)

*Get data pointer to chunk.*

  - long [l4shmc\\_chunk\\_size](#) (l4shmc\_chunk\_t \*chunk)

*Get current size of a chunk.*

  - long [l4shmc\\_chunk\\_capacity](#) (l4shmc\_chunk\_t \*chunk)

*Get capacity of a chunk.*

  - l4shmc\_signal\_t \* [l4shmc\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk)

*Get the signal of a chunk.*

  - [l4\\_cap\\_idx\\_t](#) [l4shmc\\_signal\\_cap](#) (l4shmc\_signal\_t \*signal)

*Get the signal capability of a signal.*

  - long [l4shmc\\_check\\_magic](#) (l4shmc\_chunk\_t \*chunk)

*Check magic value of a chunk.*

  - long [l4shmc\\_area\\_size](#) (l4shmc\_area\_t \*shmbarea)

*Get size of shared memory area.*

  - long [l4shmc\\_area\\_size\\_free](#) (l4shmc\_area\_t \*shmbarea)

*Get free size of shared memory area.*

  - long [l4shmc\\_area\\_overhead](#) (void)

*Get memory overhead per area that is not available for chunks.*

  - long [l4shmc\\_chunk\\_overhead](#) (void)

*Get memory overhead required in addition to the chunk capacity for adding one chunk.*

### 15.251.1 Detailed Description

Shared memory library header file.

Definition in file [shmc.h](#).

## 15.252 shmc.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #pragma once
00014
00015 #include <l4/sys/linkage.h>
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/err.h>
00018
00069 #define __INCLUDED_FROM_L4SHMC_H__
00070 #include <l4/shmc/types.h>
00071
00072 __BEGIN_DECLS
00073
00084 L4_CV long
00085 l4shmc_create(const char *shmc_name, l4_umword_t shm_size);
00086
00098 L4_CV L4_INLINE long
00099 l4shmc_attach(const char *shmc_name, l4shmc_area_t *shmarea);
00100
00113 L4_CV long
00114 l4shmc_attach_to(const char *shmc_name, l4_umword_t timeout_ms,
00115 l4shmc_area_t *shmarea);
00116
00129 L4_CV long
00130 l4shmc_add_chunk(l4shmc_area_t *shmarea,
00131 const char *chunk_name,
00132 l4_umword_t chunk_capacity,
00133 l4shmc_chunk_t *chunk);
00134
00146 L4_CV long
00147 l4shmc_add_signal(l4shmc_area_t *shmarea,
00148 const char *signal_name,
00149 l4shmc_signal_t *signal);
00150
00160 L4_CV L4_INLINE long
00161 l4shmc_trigger(l4shmc_signal_t *signal);
00162
00172 L4_CV L4_INLINE long
00173 l4shmc_chunk_try_to_take(l4shmc_chunk_t *chunk);
00174
00185 L4_CV L4_INLINE long
00186 l4shmc_chunk_ready(l4shmc_chunk_t *chunk, l4_umword_t size);
00187
00198 L4_CV L4_INLINE long
00199 l4shmc_chunk_ready_sig(l4shmc_chunk_t *chunk, l4_umword_t size);
00200
00212 L4_CV L4_INLINE long
00213 l4shmc_get_chunk(l4shmc_area_t *shmarea,
00214 const char *chunk_name,
00215 l4shmc_chunk_t *chunk);
00216
00230 L4_CV long
00231 l4shmc_get_chunk_to(l4shmc_area_t *shmarea,
00232 const char *chunk_name,
00233 l4_umword_t timeout_ms,
00234 l4shmc_chunk_t *chunk);
00235
00249 L4_CV long
00250 l4shmc_iterate_chunk(l4shmc_area_t *shmarea, const char **chunk_name,
00251 long offs);
00252
00265 L4_CV L4_INLINE long
00266 l4shmc_attach_signal(l4shmc_area_t *shmarea,
00267 const char *signal_name,
00268 l4_cap_idx_t thread,
00269 l4shmc_signal_t *signal);
00270
00285 L4_CV long
00286 l4shmc_attach_signal_to(l4shmc_area_t *shmarea,
00287 const char *signal_name,
00288 l4_cap_idx_t thread,
00289 l4_umword_t timeout_ms,
00290 l4shmc_signal_t *signal);
00291
00305 L4_CV long
00306 l4shmc_get_signal_to(l4shmc_area_t *shmarea,
00307 const char *signal_name,

```

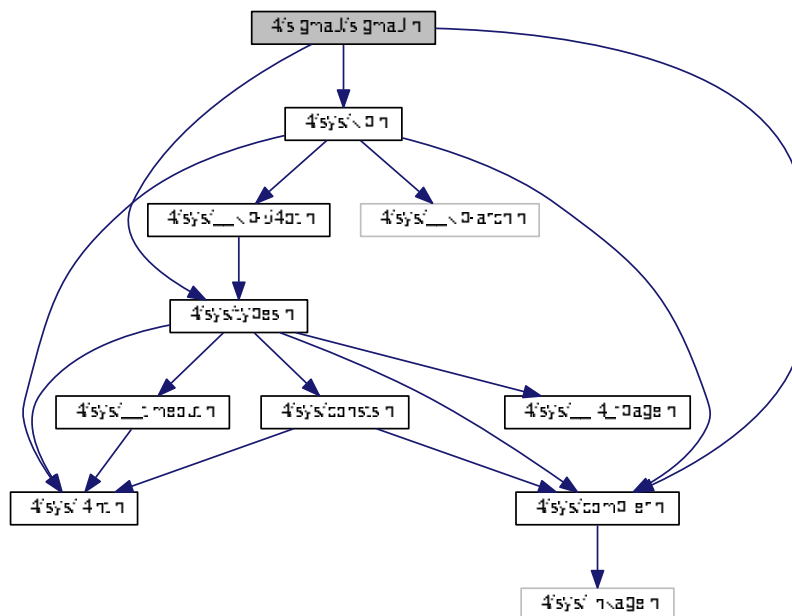
```

00308 l4_umword_t timeout_ms,
00309 l4shmc_signal_t *signal);
00310
00311 L4_CV L4_INLINE long
00312 l4shmc_get_signal(l4shmc_area_t *shmarea,
00313 const char *signal_name,
00314 l4shmc_signal_t *signal);
00315
00316
00330 L4_CV long
00331 l4shmc_enable_signal(l4shmc_signal_t *signal);
00332
00346 L4_CV long
00347 l4shmc_enable_chunk(l4shmc_chunk_t *chunk);
00348
00358 L4_CV L4_INLINE long
00359 l4shmc_wait_any(l4shmc_signal_t **retsignal);
00360
00374 L4_CV L4_INLINE long
00375 l4shmc_wait_any_try(l4shmc_signal_t **retsignal);
00376
00391 L4_CV long
00392 l4shmc_wait_any_to(l4_timeout_t timeout, l4shmc_signal_t **retsignal);
00393
00403 L4_CV L4_INLINE long
00404 l4shmc_wait_signal(l4shmc_signal_t *signal);
00405
00416 L4_CV long
00417 l4shmc_wait_signal_to(l4shmc_signal_t *signal, l4_timeout_t timeout);
00418
00432 L4_CV L4_INLINE long
00433 l4shmc_wait_signal_try(l4shmc_signal_t *signal);
00434
00444 L4_CV L4_INLINE long
00445 l4shmc_wait_chunk(l4shmc_chunk_t *chunk);
00446
00461 L4_CV long
00462 l4shmc_wait_chunk_to(l4shmc_chunk_t *chunk, l4_timeout_t timeout);
00463
00477 L4_CV L4_INLINE long
00478 l4shmc_wait_chunk_try(l4shmc_chunk_t *chunk);
00479
00489 L4_CV L4_INLINE long
00490 l4shmc_chunk_consumed(l4shmc_chunk_t *chunk);
00491
00501 L4_CV long
00502 l4shmc_connect_chunk_signal(l4shmc_chunk_t *chunk,
00503 l4shmc_signal_t *signal);
00504
00514 L4_CV L4_INLINE long
00515 l4shmc_is_chunk_ready(l4shmc_chunk_t *chunk);
00516
00526 L4_CV L4_INLINE long
00527 l4shmc_is_chunk_clear(l4shmc_chunk_t *chunk);
00528
00538 L4_CV L4_INLINE void *
00539 l4shmc_chunk_ptr(l4shmc_chunk_t *chunk);
00540
00550 L4_CV L4_INLINE long
00551 l4shmc_chunk_size(l4shmc_chunk_t *chunk);
00552
00562 L4_CV L4_INLINE long
00563 l4shmc_chunk_capacity(l4shmc_chunk_t *chunk);
00564
00574 L4_CV L4_INLINE l4shmc_signal_t *
00575 l4shmc_chunk_signal(l4shmc_chunk_t *chunk);
00576
00585 L4_CV L4_INLINE l4_cap_idx_t
00586 l4shmc_signal_cap(l4shmc_signal_t *signal);
00587
00597 L4_CV L4_INLINE long
00598 l4shmc_check_magic(l4shmc_chunk_t *chunk);
00599
00609 L4_CV L4_INLINE long
00610 l4shmc_area_size(l4shmc_area_t *shmarea);
00611
00623 L4_CV long
00624 l4shmc_area_size_free(l4shmc_area_t *shmarea);
00625
00632 L4_CV long
00633 l4shmc_area_overhead(void);
00634
00642 L4_CV long
00643 l4shmc_chunk_overhead(void);
00644
00645 #include <l4/shmc/internal.h>
00646

```

### 15.253 I4/sigma0/sigma0.h File Reference

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
Include dependency graph for sigma0.h:
```



- #define SIGMA0\_REQ\_MAGIC ~0xFFUL  
*Request magic.*
- #define SIGMA0\_REQ\_MASK ~0xFFUL  
*Request mask.*
- #define SIGMA0\_REQ\_ID\_MASK 0xF0  
*ID mask.*
- #define SIGMA0\_REQ\_ID\_FPAGE\_RAM 0x60  
*RAM.*
- #define SIGMA0\_REQ\_ID\_FPAGE\_IOMEM 0x70  
*I/O memory.*
- #define SIGMA0\_REQ\_ID\_FPAGE\_IOMEM\_CACHED 0x80  
*Cached I/O memory.*
- #define SIGMA0\_REQ\_ID\_FPAGE\_ANY 0x90  
*Any.*

- `#define SIGMA0_REQ_ID_KIP 0xA0`  
*KIP.*
- `#define SIGMA0_REQ_ID_TBUF 0xB0`  
*TBUF.*
- `#define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0`  
*Debug dump.*
- `#define SIGMA0_REQ_ID_NEW_CLIENT 0xD0`  
*New client.*
- `#define SIGMA0_IS_MAGIC_REQ(d1) ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)`  
*Check if magic.*
- `#define SIGMA0_REQ(x) (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)`  
*Construct.*
- `#define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))`  
*RAM.*
- `#define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))`  
*I/O memory.*
- `#define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))`  
*Cache I/O memory.*
- `#define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))`  
*Any.*
- `#define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))`  
*KIP.*
- `#define SIGMA0_REQ_TBUF (SIGMA0_REQ(TBUF))`  
*TBUF.*
- `#define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))`  
*Debug dump.*
- `#define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))`  
*New client.*

## Enumerations

- `enum l4sigma0_return_flags_t {`  
`L4SIGMA0_OK, L4SIGMA0_NOTALIGNED, L4SIGMA0_IPCERROR, L4SIGMA0_NOFPAGE ,`  
`L4SIGMA0_SMALLERFPAGE }`  
*Return flags of libsigma0 functions.*

## Functions

- `l4_kernel_info_t * l4sigma0_map_kip (l4_cap_idx_t sigma0, void *addr, unsigned log2_size)`  
*Map the kernel info page from pager to addr.*
- `int l4sigma0_map_mem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size)`  
*Request a memory mapping from sigma0.*
- `int l4sigma0_map_iomem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size, int cached)`  
*Request IO memory from sigma0.*
- `int l4sigma0_map_anypage (l4_cap_idx_t sigma0, l4_addr_t map_area, unsigned log2_map_size, l4_addr_t *base, unsigned sz)`  
*Request an arbitrary free page of RAM.*
- `int l4sigma0_map_tbuf (l4_cap_idx_t sigma0, l4_addr_t virt)`  
*Request Fiasco trace buffer.*
- `void l4sigma0_debug_dump (l4_cap_idx_t sigma0)`

*Request sigma0 to dump internal debug information.*

- `int l4sigma0_new_client (l4_cap_idx_t sigma0, l4_cap_idx_t gate)`

*Create a new IPC gate for a new Sigma0 client.*

- `char const * l4sigma0_map_errstr (int err)`

*Get a user readable error messages for the return codes.*

## 15.253.1 Detailed Description

Sigma0 interface.

Definition in file [sigma0.h](#).

## 15.254 sigma0.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015 #ifndef __L4_SIGMA0_SIGMA0_H
00016 #define __L4_SIGMA0_SIGMA0_H
00017
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/kip.h>
00029
00036 #undef SIGMA0_REQ_MAGIC
00037 #undef SIGMA0_REQ_MASK
00038
00039 # define SIGMA0_REQ_MAGIC ~0xFFFUL
00040 # define SIGMA0_REQ_MASK ~0xFFFUL
00042 /* Starting with 0x60 allows to detect components which still use the old
00043 * constants (0x00 ... 0x50) */
00044 #define SIGMA0_REQ_ID_MASK 0xF0
00045 #define SIGMA0_REQ_ID_FPAGE_RAM 0x60
00046 #define SIGMA0_REQ_ID_FPAGE_IOMEM 0x70
00047 #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED 0x80
00048 #define SIGMA0_REQ_ID_FPAGE_ANY 0x90
00049 #define SIGMA0_REQ_ID_KIP 0xA0
00050 #define SIGMA0_REQ_ID_TBUF 0xB0
00051 #define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0
00052 #define SIGMA0_REQ_ID_NEW_CLIENT 0xD0
00054 #define SIGMA0_IS_MAGIC_REQ(d1) \
00055 ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
00057 #define SIGMA0_REQ(x) \
00058 (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
00060 /* Use these constants in your code! */
00061 #define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))
00062 #define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))
00063 #define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
00064 #define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))
00065 #define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))
00066 #define SIGMA0_REQ_TBUF (SIGMA0_REQ(TBUF))
00067 #define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))
00068 #define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))
00070
00071
00075
00079 enum l4sigma0_return_flags_t {
00080 L4SIGMA0_OK,
00081 L4SIGMA0_NOTALIGNED,
00082 L4SIGMA0_IPCERROR,
00083 L4SIGMA0_NOFPAGE,
00084 L4SIGMA0_4,
00085 L4SIGMA0_5,
00086 L4SIGMA0_SMALLERFPAGE,
00087 };
00088

```

```

00089 EXTERN_C_BEGIN
00090
00099 L4_CV l4_kernel_info_t *
00100 l4sigma0_map_kip(l4_cap_idx_t sigma0, void *addr, unsigned log2_size);
00101
00115 L4_CV int l4sigma0_map_mem(l4_cap_idx_t sigma0,
00116 l4_addr_t phys, l4_addr_t virt,
00117 l4_addr_t size);
00117
00141 L4_CV int l4sigma0_map_iomem(l4_cap_idx_t sigma0,
00142 l4_addr_t phys,
00143 l4_addr_t virt, l4_addr_t size, int cached);
00163 L4_CV int l4sigma0_map_anypage(l4_cap_idx_t sigma0,
00164 l4_addr_t map_area,
00165 unsigned log2_map_size, l4_addr_t *base,
00166 unsigned sz);
00179 L4_CV int l4sigma0_map_tbuf(l4_cap_idx_t sigma0,
00180 l4_addr_t virt);
00180
00189 L4_CV void l4sigma0_debug_dump(l4_cap_idx_t sigma0);
00190
00196 L4_CV int l4sigma0_new_client(l4_cap_idx_t sigma0,
00197 l4_cap_idx_t gate);
00197
00204 L4_INLINE char const *l4sigma0_map_errstr(int err);
00205
00209 /* Implementations */
00210
00211 L4_INLINE char const *l4sigma0_map_errstr(int err)
00212 {
00213 switch (err)
00214 {
00215 case 0: return "No error";
00216 case -1: return "Phys, virt or size not aligned";
00217 case -2: return "IPC error";
00218 case -3: return "No fpage received";
00219 #ifndef SIGMA0_REQ_MAGIC
00220 case -4: return "Bad physical address (old protocol only)";
00221 #endif
00222 case -6: return "Superpage requested but smaller flexpage received";
00223 case -7: return "Cannot map I/O memory cacheable (old protocol only)";
00224 default: return "Unknown error";
00225 }
00226 }
00227
00228 EXTERN_C_END
00229
00231 #endif /* ! __L4_SIGMA0_SIGMA0_H */

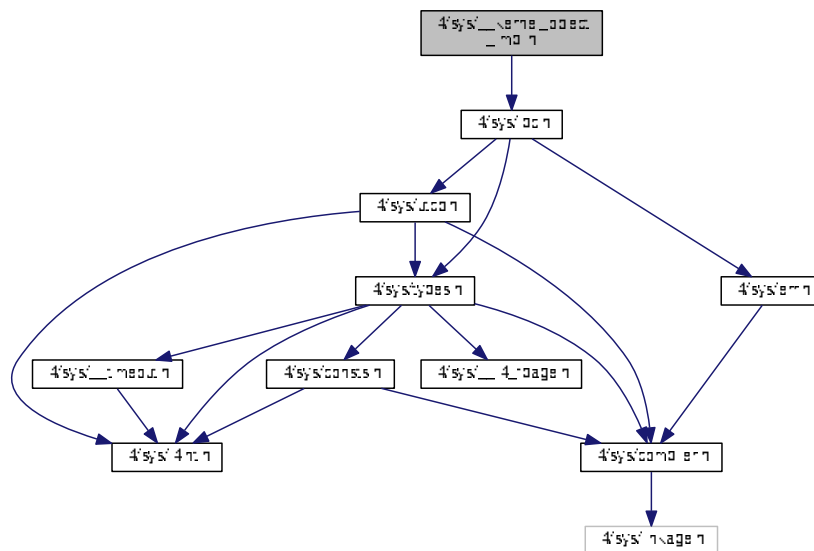
```

## 15.255 l4/sys/\_\_kernel\_object\_impl.h File Reference

Low-level kernel debugger functions.



Include dependency graph for `__kernel_object_impl.h`:

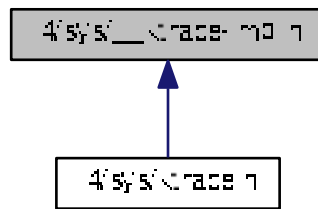
[illegible]

Definition in file [kernel\\_object\\_impl.h](#).

```
00001
00005 #pragma once
00006
00007 #include <l4/sys/ipc.h>
00008
00009 L4_INLINE l4_msgtag_t
00010 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag,
00011 l4_utcb_t *utcb) L4_NOTHROW
00012 {
00013 l4_msgtag_t t2;
00014 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00015
00016 if (l4_is_invalid_cap(obj))
00017 return l4_msgtag(-L4_EINVAL, 0, 0, 0);
```



This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_tracebuffer\\_status\\_t \\* fiasco\\_tbuf\\_get\\_status](#) (void)  
*Return trace-buffer status.*
- [l4\\_addr\\_t fiasco\\_tbuf\\_get\\_status\\_phys](#) (void)  
*Return the physical address of the trace-buffer status struct.*
- [l4\\_umword\\_t fiasco\\_tbuf\\_log](#) (const char \*text)  
*Create new trace-buffer entry with describing <text>.*
- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_3val](#) (const char \*text, [l4\\_umword\\_t](#) v1, [l4\\_umword\\_t](#) v2, [l4\\_umword\\_t](#) v3)  
*Create new trace-buffer entry with describing <text> and three additional values.*
- void [fiasco\\_tbuf\\_clear](#) (void)  
*Clear trace-buffer.*
- void [fiasco\\_tbuf\\_dump](#) (void)  
*Dump trace-buffer to kernel console.*
- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_binary](#) (const unsigned char \*data)  
*Create new trace-buffer entry with binary data.*

### 15.257.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [\\_\\_ktrace-impl.h](#).

## 15.258 \_\_ktrace-impl.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate

```

```

00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/kdebug.h>
00029
00030 /*****
00031 *** Implementation
00032 *****/
00033
00034 L4_INLINE l4_tracebuffer_status_t *
00035 fiasco_tbuf_get_status(void)
00036 {
00037 return 0;
00038 /* Not implemented */
00039 }
00040
00041 L4_INLINE l4_addr_t
00042 fiasco_tbuf_get_status_phys(void)
00043 {
00044 return ~0UL;
00045 }
00046
00047 L4_INLINE l4_umword_t
00048 fiasco_tbuf_log(const char *text)
00049 {
00050 enum { TBUF_LOG = 0x201 };
00051 return l4_error(__kdebug_text(TBUF_LOG, text, __builtin_strlen(text)));
00052 }
00053
00054 L4_INLINE l4_umword_t
00055 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1,
00056 l4_umword_t v2,
00057 l4_umword_t v3)
00058 {
00059 enum { TBUF_LOG_3VAL = 0x204 };
00060 return l4_error(__kdebug_3_text(TBUF_LOG_3VAL, text,
00061 __builtin_strlen(text), v1, v2, v3));
00062 }
00063 L4_INLINE void
00064 fiasco_tbuf_clear(void)
00065 {
00066 enum { TBUF_CLEAR = 0x202 };
00067 __kdebug_op(TBUF_CLEAR);
00068 }
00069
00070 L4_INLINE void
00071 fiasco_tbuf_dump(void)
00072 {
00073 enum { TBUF_DUMP = 0x203 };
00074 __kdebug_op(TBUF_DUMP);
00075 }
00076
00077 L4_INLINE l4_umword_t
00078 fiasco_tbuf_log_binary(const unsigned char *data)
00079 {
00080 enum { TBUF_LOG_BIN = 0x208 };
00081 return l4_error(__kdebug_text(TBUF_LOG_BIN, (const char *)data, 24));
00082 }
00083

```

## 15.259 l4/sys/\_\_typeinfo.h File Reference

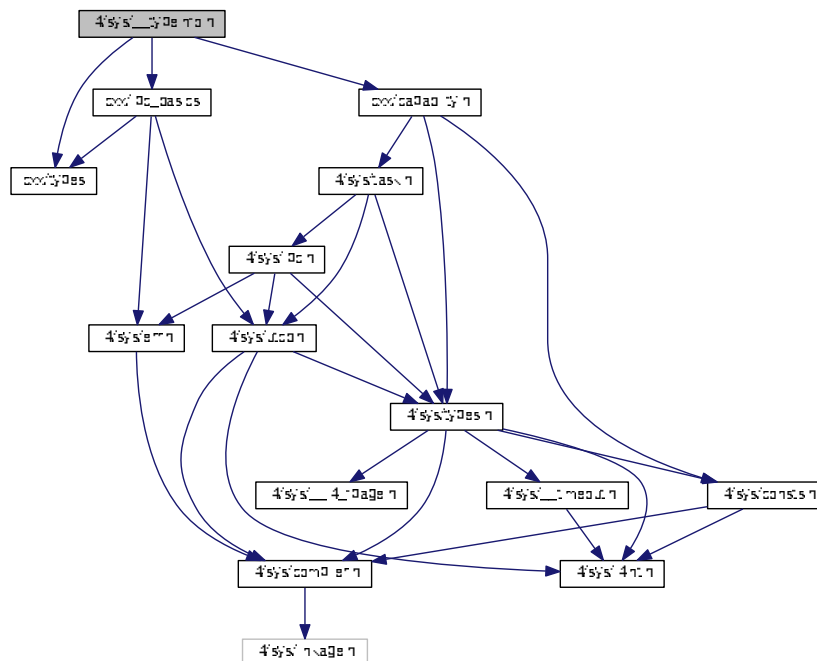
Type information handling.

```

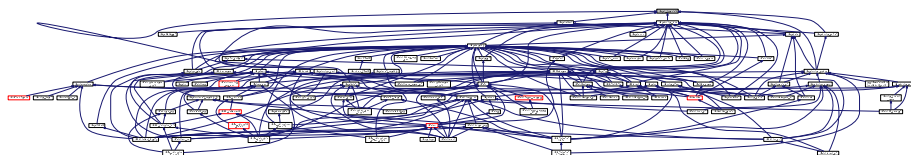
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"

```

Include dependency graph for \_\_typeinfo.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `L4::Typeid::P_dispatch< LIST >`  
*Use for protocol based dispatch stage.*
- struct `L4::Typeid::Detail::Rpc_end`  
*Internal end-of-list marker.*
- struct `L4::Typeid::Detail::_Rpc< OPCODE, O, X >`  
*Empty list of RPCs.*
- struct `L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >`  
*Non-empty list of RPCs.*
- struct `L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >`  
*Find the given RPC in the list.*
- struct `L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >`  
*Find the given RPC in the list.*
- struct `L4::Typeid::Raw_ipc< CLASS >`  
*RPCs list for passing raw incoming IPC to the server object.*

- struct [L4::Typeid::Rpc< RPCS >](#)  
*Standard list of RPCs of an interface.*
- struct [L4::Typeid::Rpc\\_code< OPCODE\\_TYPE >](#)  
*List of RPCs of an interface using a special opcode type.*
- struct [L4::Typeid::Rpc\\_code< OPCODE\\_TYPE >::F< RPCS >](#)
- struct [L4::Typeid::Rpc\\_nocode< OPERATION >](#)  
*List of RPCs of an interface using a single operation without an opcode.*
- struct [L4::Typeid::Rpc\\_sys< ARG >](#)  
*List of RPCs typically used for kernel interfaces.*
- struct [L4::Type\\_info](#)  
*Dynamic Type Information for [L4Re](#) Interfaces.*
- class [L4::Type\\_info::Demand](#)  
*Data type for expressing the needed receive buffers at the server-side of an interface.*
- struct [L4::Type\\_info::Demand\\_t< CAPS, FLAGS, MEM, PORTS >](#)  
*Template type statically describing demand of receive buffers.*
- struct [L4::Type\\_info::Demand\\_union\\_t< D1, D2 >](#)  
*Template type statically describing the combination of two [Demand](#) object.*
- struct [L4::Kobject\\_typeid< T >](#)  
*Meta object for handling access to type information of Kobjects.*
- struct [L4::Kobject\\_typeid< void >](#)  
*Minimalistic ID for `void` interface.*
- class [L4::Kobject\\_t< Derived, Base, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from a single base class.*
- class [L4::Kobject\\_2t< Derived, Base1, Base2, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_3t< Derived, Base1, Base2, Base3, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_demand< T >](#)  
*Get the combined server-side resource requirements for all type `T...`*
- struct [L4::Proto\\_t< P >](#)  
*Data type for defining protocol numbers.*
- struct [L4::Kobject\\_x< Derived, ARGS >](#)  
*Generic [Kobject](#) inheritance template.*

## Namespaces

- [L4](#)  
*[L4](#) low-level kernel interface.*
- [L4::Typeid](#)  
*Definition of interface data-type helpers.*

## Typedefs

- typedef int [L4::Opcode](#)  
*Data type for RPC opcodes.*

## Enumerations

- enum { [L4::PROTO\\_ANY](#) = 0, [L4::PROTO\\_EMPTY](#) = -19 }

## Functions

- `template<typename T>`  
`Type_info const * L4::kobject_typeid ()`  
*Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in *T*.*

### 15.259.1 Detailed Description

Type information handling.

Definition in file [\\_\\_typeinfo.h](#).

## 15.260 \_\_typeinfo.h

```

00001
00005 /*
00006 * Copyright (C) 2015 Kernkonzept GmbH.
00007 * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008 */
00009 /*
00010 * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027 #pragma GCC system_header
00028
00029 #include "cxx/types"
00030 #include "cxx/ipc_basics"
00031 #include "cxx/capability.h"
00032
00033 #if defined(__GXX_RTTI) && !defined(L4_NO_RTTI)
00034 # include <typeinfo>
00035 typedef std::type_info const *L4_std_type_info_ptr;
00036 # define L4_KOBJECT_META_RTTI(type) (&typeid(type))
00037 inline char const *L4_kobject_type_name(L4_std_type_info_ptr n)
00038 { return n ? n->name() : 0; }
00039 #else
00040 typedef void const *L4_std_type_info_ptr;
00041 # define L4_KOBJECT_META_RTTI(type) (0)
00042 inline char const *L4_kobject_type_name(L4_std_type_info_ptr)
00043 { return 0; }
00044 #endif
00045
00046 namespace L4 {
00047 typedef int Opcode;
00048 // internal max helpers
00049 namespace __I {
00050 // internal max of A and B helper
00051 template< unsigned char A, unsigned char B>
00052 struct Max { enum { Res = A > B ? A : B }; };
00053 } // namespace __I
00054
00055 enum
00056 {
00058 PROTO_ANY = 0,
00060 PROTO_EMPTY = -19,
00061 };
00062
00077 namespace Typeid {
00083 using namespace L4::Types;
00084

```

```

00085 /*****/
00093 template<long P, typename T>
00094 struct Iface
00095 {
00096 typedef Iface type;
00097 typedef T iface_type;
00098 enum { Proto = P };
00099 };
00100
00101
00102 /*****/
00108 struct Iface_list_end
00109 {
00110 typedef Iface_list_end type;
00111 static bool contains(long) { return false; }
00112 };
00113
00114
00122 template<typename I, typename N = Iface_list_end>
00123 struct Iface_list
00124 {
00125 typedef Iface_list<I, N> type;
00126
00127 typedef typename I::iface_type iface_type;
00128 typedef N Next;
00129
00130 enum { Proto = I::Proto };
00131
00132 static bool contains(long proto)
00133 { return (proto == Proto) || Next::contains(proto); }
00134 };
00135
00136 // do not insert PROTO_EMPTY interfaces
00137 template<typename I, typename N>
00138 struct Iface_list<Iface<PROTO_EMPTY, I>, N> : N {};
00139
00140 // do not insert 'void' type interfaces
00141 template<long P, typename N>
00142 struct Iface_list<Iface<P, void>, N> : N {};
00143
00144
00145 /*****/
00146 /*
00147 * \internal
00148 * Test if an interface I is in list L
00149 * \tparam I Interface for lookup
00150 * \tparam L Iface_list for search
00151 */
00152 template< typename I, typename L >
00153 struct _In_list;
00154
00155 template< typename I >
00156 struct _In_list<I, Iface_list_end> : False {};
00157
00158 template< typename I, typename N >
00159 struct _In_list<I, Iface_list<I, N> > : True {};
00160
00161 template< typename I, typename I2, typename N >
00162 struct _In_list<I, Iface_list<I2, N> > : _In_list<I, typename N::type> {};
00163
00164 template<typename I, typename L>
00165 struct In_list : _In_list<typename I::type, typename L::type> {};
00166
00167
00168 /*****/
00169 /*
00170 * \internal
00171 * Add Helper: add I to interface list L if ADD is true
00172 * \ingroup l4_cxx_ipc_internal
00173 */
00174 template< bool ADD, typename I, typename L>
00175 struct _Iface_list_add;
00176
00177 template< typename I, typename L>
00178 struct _Iface_list_add<false, I, L> : L {};
00179
00180 template< typename I, typename L>
00181 struct _Iface_list_add<true, I, L> : Iface_list<I, L> {};
00182
00183 /*
00184 * \internal
00185 * Add Helper: add I to interface list L if not already in L.
00186 * \ingroup l4_cxx_ipc_internal
00187 */
00188 template< typename I, typename L >
00189 struct Iface_list_add :
00190 _Iface_list_add<

```



```

00191 !In_list<I, typename L::type>::value, I, typename L::type>
00192 {});
00193
00194 /*****
00195 */
00196 * \internal
00197 * Helper: checking for a conflict between I2 and I2.
00198 * A conflict means I1 and I2 have the same protocol ID but a different
00199 * iface_type.
00200 */
00201 template< typename I1, typename I2 >
00202 struct __Iface_conflict : Bool<I1::Proto != PROTO_EMPTY && I1::Proto == I2::Proto> {};
00203
00204 template< typename I >
00205 struct __Iface_conflict<I, I> : False {};
00206
00207 /
00208 * \internal
00209 * Helper: checking for a conflict between I and any interface in LIST.
00210 */
00211 template< typename I, typename LIST >
00212 struct _Iface_conflict;
00213
00214 template< typename I >
00215 struct _Iface_conflict<I, Iface_list_end> : False {};
00216
00217 template< typename I, typename I2, typename LIST >
00218 struct _Iface_conflict<I, Iface_list<I2, LIST> > :
00219 Bool<__Iface_conflict<I, I2>::value || _Iface_conflict<I, typename LIST::type>::value>
00220 {};
00221
00222 template< typename I, typename LIST >
00223 struct Iface_conflict : _Iface_conflict<typename I::type, typename LIST::type> {};
00224
00225 /*****
00226 */
00227 * \internal
00228 * Helper: merge two interface lists
00229 */
00230 template< typename L1, typename L2 >
00231 struct _Merge_list;
00232
00233 template< typename L >
00234 struct _Merge_list<Iface_list_end, L> : L {};
00235
00236 template< typename I, typename L1, typename L2 >
00237 struct _Merge_list<Iface_list<I, L1>, L2> :
00238 _Merge_list<typename L1::type, typename Iface_list_add<I, L2>::type> {};
00239
00240 template<typename L1, typename L2>
00241 struct Merge_list : _Merge_list<typename L1::type, typename L2::type> {};
00242
00243 /*****
00244 */
00245 * \internal
00246 * check for conflicts among all interfaces in L1 with any interfaces in L2.
00247 */
00248 template< typename L1, typename L2 >
00249 struct _Conflict;
00250
00251 template< typename L >
00252 struct _Conflict<Iface_list_end, L> : False {};
00253
00254 template< typename I, typename L1, typename L2 >
00255 struct _Conflict<Iface_list<I, L1>, L2> :
00256 Bool<Iface_conflict<I, typename L2::type>::value
00257 || _Conflict<typename L1::type, typename L2::type>::value> {};
00258
00259 template< typename L1, typename L2 >
00260 struct Conflict : _Conflict<typename L1::type, typename L2::type> {};
00261
00262 // to be removed -----
00263 // p_dispatch code -- for legacy dispatch -----
00264 /*****
00265 */
00266 * \internal
00267 * helper: Dispatch helper for calling server-side p_dispatch() functions.
00268 */
00269 template<typename LIST>
00270 struct _P_dispatch;
00271
00272 // No matching dispatcher found
00273 template<>
00274 struct _P_dispatch<Iface_list_end>
00275 {
00276 template< typename THIS, typename A1, typename A2 >
00277 static int f(THIS *, long, A1, A2 &)

```

```

00282 { return -L4_EBADPROTO; }
00283 };
00284
00285
00286 // call matching p_dispatch() function
00287 template< typename I, typename LIST >
00288 struct _P_dispatch<Iface_list<I, LIST> >
00289 {
00290 // special handling for the meta protocol, to avoid 'using' murx
00291 template< typename THIS, typename A1, typename A2 >
00292 static int _f(THIS self, A1, A2 &a2, True::type)
00293 {
00294 return self->dispatch_meta_request(a2);
00295 }
00296
00297 // normal p_dispatch() dispatching
00298 template< typename THIS, typename A1, typename A2 >
00299 static int _f(THIS self, A1 a1, A2 &a2, False::type)
00300 {
00301 return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
00302 a1, a2);
00303 }
00304
00305 // dispatch function with switch for meta protocol
00306 template< typename THIS, typename A1, typename A2 >
00307 static int f(THIS *self, long proto, A1 a1, A2 &a2)
00308 {
00309 if (I::Proto == proto)
00310 return _f(self, a1, a2, Bool<I::Proto == (long)L4_PROTO_META>());
00311
00312 return _P_dispatch<typename LIST::type>::f(self, proto, a1, a2);
00313 }
00314 };
00315
00316 template<typename LIST>
00317 struct P_dispatch : _P_dispatch<typename LIST::type> {};
00318 // end: p_dispatch -----
00319 // end: to be removed -----
00320
00321
00322 template<typename RPC> struct Default_op;
00323
00324 namespace Detail {
00325
00326 struct Rpcs_end
00327 {
00328 typedef void opcode_type;
00329 typedef Rpcs_end rpc;
00330 typedef Rpcs_end type;
00331 };
00332 };
00333
00334 template<typename O1, typename O2, typename RPCS>
00335 struct _Rpc : _Rpc<typename RPCS::next::rpc, O2, typename RPCS::next::type> {};
00336
00337 template<typename O1, typename O2>
00338 struct _Rpc<O1, O2, Rpcs_end> {};
00339
00340 template<typename OP, typename RPCS>
00341 struct _Rpc<OP, OP, RPCS> : RPCS
00342 {
00343 typedef _Rpc type;
00344 };
00345
00346 template<typename OP, typename RPCS>
00347 struct Rpc : _Rpc<typename RPCS::rpc, OP, RPCS> {};
00348
00349 template<typename T, unsigned CODE>
00350 struct _Get_opcode
00351 {
00352 template<bool, typename> struct Invalid_opcode {};
00353 template<typename X> struct Invalid_opcode<true, X>;
00354
00355 private:
00356 template<typename U, U> struct _chk;
00357 template<typename U> static long _opc(_chk<int, U::Opcode> *);
00358 template<typename U> static char _opc(...);
00359
00360 template<unsigned SZ, typename U>
00361 struct _Opc { enum { value = CODE }; };
00362
00363 template<typename U>
00364 struct _Opc<sizeof(long), U> { enum { value = U::Opcode }; };
00365
00366 public:
00367 enum { value = _Opc<sizeof(_opc<T>)(0), T::value };
00368 Invalid_opcode<(value < CODE), T> invalid_opcode;
00369 };
00370
00371
00372

```

```

00374 template<typename OPCODE, unsigned O, typename ...X>
00375 struct _Rpcs : Rpcs_end {};
00376
00377 template<typename OPCODE, unsigned O, typename R, typename ...X>
00378 struct _Rpcs<OPCODE, O, R, X...>
00379 {
00380 {
00381 typedef _Rpcs type;
00382 typedef OPCODE opcode_type;
00383 typedef R rpc;
00384 typedef typename _Rpcs<OPCODE, _Get_opcode<R, O>::value + 1, X...
00385 >::type next;
00386 {
00387 enum { Opcode = _Get_opcode<R, O>::value };
00388 template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpcs> {};
00389 };
00390
00391 template<typename OPCODE, unsigned O, typename R>
00392 struct _Rpcs<OPCODE, O, Default_op<R> >
00393 {
00394 {
00395 typedef _Rpcs type;
00396 typedef void opcode_type;
00397 typedef R rpc;
00398 typedef Rpcs_end next;
00399 enum { Opcode = -99 };
00400 template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpcs> {};
00401 };
00402
00403 } // namespace Detail
00404
00405 template<typename CLASS>
00406 struct Raw_ipc
00407 {
00408 {
00409 typedef Raw_ipc type;
00410 typedef Detail::Rpcs_end next;
00411 typedef void opcode_type;
00412 };
00413
00414 template<typename ...RPCS>
00415 struct Rpcs : Detail::_Rpcs<L4::Opcode, 0, RPCS...> {};
00416
00417 template<typename OPCODE_TYPE>
00418 struct Rpcs_code
00419 {
00420 {
00421 template<typename ...RPCS>
00422 struct F : Detail::_Rpcs<OPCODE_TYPE, 0, RPCS...> {};
00423 };
00424
00425 template<typename OPERATION>
00426 struct Rpc_nocode : Detail::_Rpcs<void, 0, OPERATION> {};
00427
00428 template<typename ...ARG>
00429 struct Rpcs_sys : Detail::_Rpcs<l4_umword_t, 0, ARG...> {};
00430
00431 template<typename CLASS>
00432 struct Rights
00433 {
00434 {
00435 unsigned rights;
00436 Rights(unsigned rights) : rights(rights) {}
00437 unsigned operator & (unsigned rhs) const { return rights & rhs; }
00438 };
00439
00440 } // namespace Typeid
00441
00442 struct L4_EXPORT Type_info
00443 {
00444 {
00445 class L4_EXPORT Demand
00446 {
00447 {
00448 private:
00449 static unsigned char max(unsigned char a, unsigned char b)
00450 { return a > b ? a : b; }
00451
00452 public:
00453 unsigned char caps;
00454 unsigned char flags;
00455 unsigned char mem;
00456 unsigned char ports;
00457
00458 explicit
00459 Demand(unsigned char caps = 0, unsigned char flags = 0,
00460 unsigned char mem = 0, unsigned char ports = 0)
00461 : caps(caps), flags(flags), mem(mem), ports(ports) {}
00462
00463 bool no_demand() const
00464 { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }
00465
00466 Demand operator | (Demand const &rhs) const
00467 {
00468 return Demand(max(caps, rhs.caps), flags | rhs.flags,

```

```

00549 max(mem, rhs.mem), max(ports, rhs.ports));
00550 }
00551 };
00552
00561 template<unsigned char CAPS = 0, unsigned char FLAGS = 0,
00562 unsigned char MEM = 0, unsigned char PORTS = 0>
00563 struct Demand_t : Demand
00564 {
00565 enum
00566 {
00567 Caps = CAPS,
00568 Flags = FLAGS,
00569 Mem = MEM,
00570 Ports = PORTS
00571 };
00572 Demand_t() : Demand(CAPS, FLAGS, MEM, PORTS) {}
00573 };
00574
00582 template<typename D1, typename D2>
00583 struct Demand_union_t : Demand_t<__I::Max<D1::Caps, D2::Caps>::Res,
00584 D1::Flags | D2::Flags,
00585 __I::Max<D1::Mem, D2::Mem>::Res,
00586 __I::Max<D1::Ports, D2::Ports>::Res>
00587 {};
00588
00589 L4_std_type_info_ptr _type;
00590 Type_info const *const *_bases;
00591 unsigned _num_bases;
00592 long _proto;
00593
00594 L4_std_type_info_ptr type() const { return _type; }
00595 Type_info const *base(unsigned idx) const { return _bases[idx]; }
00596 unsigned num_bases() const { return _num_bases; }
00597 long proto() const { return _proto; }
00598 char const *name() const { return L4_kobject_type_name(type()); }
00599 bool has_proto(long proto) const
00600 {
00601 if (_proto && _proto == proto)
00602 return true;
00603
00604 if (!proto)
00605 return false;
00606
00607 for (unsigned i = 0; i < _num_bases; ++i)
00608 if (base(i)->has_proto(proto))
00609 return true;
00610
00611 return false;
00612 }
00613 };
00614
00620 template<typename T> struct Kobject_typeid
00621 {
00632 typedef typename T::__Kobject_typeid::Demand Demand;
00633 typedef typename T::__Iface::iface_type Iface;
00634 typedef typename T::__Iface_list Iface_list;
00635
00640 static Type_info const *id() { return &T::__Kobject_typeid::_m; }
00641
00649 static Type_info::Demand demand()
00650 { return T::__Kobject_typeid::Demand(); }
00651
00652 // to be removed -----
00653 // p_dispatch -----
00669 template<typename THIS, typename A1, typename A2>
00670 static int proto_dispatch(THIS *self, long proto, A1 a1, A2 &a2)
00671 { return Typeid::P_dispatch<typename T::__Iface_list>::f(
00672 self, proto, a1, a2); }
00672 // p_dispatch -----
00673 // end: to be removed -----
00674 };
00675
00677 template<> struct Kobject_typeid<void>
00678 {
00679 typedef Type_info::Demand_t<> Demand;
00680 };
00681
00690 template<typename T>
00691 inline
00692 Type_info const *kobject_typeid()
00693 { return Kobject_typeid<T>::id(); }
00694
00699 #define L4___GEN_TI(t...)
00700 Type_info const t::__Kobject_typeid::_m =
00701 {
00702 L4_KOBJECT_META_RTTI(Derived),
00703 &t::__Kobject_typeid::_b[0],

```

```

00704 sizeof(t::__Kobject_typeid::_b) / sizeof(t::__Kobject_typeid::_b[0]), \
00705 PROTO \
00706 }
00707
00712 #define L4____GEN_TI_MEMBERS(BASE_DEMAND...) \
00713 private: \
00714 template< typename T > friend struct Kobject_typeid; \
00715 protected: \
00716 struct __Kobject_typeid { \
00717 typedef Type_info::Demand_union_t<S_DEMAND, BASE_DEMAND> Demand; \
00718 static Type_info const *const _b[]; \
00719 static Type_info const _m; \
00720 }; \
00721 public: \
00722 static long const Protocol = PROTO; \
00723 typedef L4::Typeid::Rights<Class> Rights; \
00724
00753 template< \
00754 typename Derived, \
00755 typename Base, \
00756 long PROTO = PROTO_ANY, \
00757 typename S_DEMAND = Type_info::Demand_t<> \
00758 > \
00759 class Kobject_t : public Base \
00760 { \
00761 protected: \
00763 typedef Derived Class; \
00765 typedef Typeid::Iface<PROTO, Derived> __Iface; \
00767 typedef Typeid::Merge_list< \
00768 Typeid::Iface_list<__Iface>, typename Base::__Iface_list \
00769 > __Iface_list; \
00770 \
00772 static void __check_protocols__() \
00773 { \
00774 typedef Typeid::Iface_conflict<__Iface, typename Base::__Iface_list> Base_conflict; \
00775 static_assert(!Base_conflict::value, "ambiguous protocol ID: protocol also used by Base"); \
00776 } \
00777 \
00779 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); } \
00780 \
00781 // Generate the remaining type information \
00782 L4____GEN_TI_MEMBERS(typename Base::__Kobject_typeid::Demand) \
00783 }; \
00784 \
00785 \
00786 template< typename Derived, typename Base, long PROTO, typename S_DEMAND> \
00787 Type_info const *const \
00788 Kobject_t<Derived, Base, PROTO, S_DEMAND>:: \
00789 __Kobject_typeid::_b[] = { &Base::__Kobject_typeid::_m }; \
00790 \
00795 template< typename Derived, typename Base, long PROTO, typename S_DEMAND> \
00796 L4____GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>); \
00797 \
00798 \
00828 template< \
00829 typename Derived, \
00830 typename Base1, \
00831 typename Base2, \
00832 long PROTO = PROTO_ANY, \
00833 typename S_DEMAND = Type_info::Demand_t<> \
00834 > \
00835 class Kobject_2t : public Base1, public Base2 \
00836 { \
00837 protected: \
00839 typedef Derived Class; \
00841 typedef Typeid::Iface<PROTO, Derived> __Iface; \
00843 typedef Typeid::Merge_list< \
00844 Typeid::Iface_list<__Iface>, \
00845 Typeid::Merge_list< \
00846 typename Base1::__Iface_list, \
00847 typename Base2::__Iface_list \
00848 > \
00849 > __Iface_list; \
00850 \
00852 static void __check_protocols__() \
00853 { \
00854 typedef typename Base1::__Iface_list Base1_proto_list; \
00855 typedef typename Base2::__Iface_list Base2_proto_list; \
00856 \
00857 typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict; \
00858 typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict; \
00859 static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1"); \
00860 static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2"); \
00861 \
00862 typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict; \
00863 static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes"); \
00864 } \

```

```

00865
00866 // disambiguate cap()
00867 l4_cap_idx_t cap() const throw()
00868 { return Base1::cap(); }
00869
00871 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); }
00872
00873 L4_____GEN_TI_MEMBERS(Type_info::Demand_union_t<
00874 typename Base1::__Kobject_typeid::Demand,
00875 typename Base2::__Kobject_typeid::Demand>
00876)
00877
00878 public:
00879 // Provide non-ambiguous conversion to Kobject
00880 operator Kobject const & () const
00881 { return *static_cast<Base1 const *>(this); }
00882
00883 // Provide non-ambiguous access of dec_refcnt()
00884 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb =
00885 l4_utcb())
00886 { return Base1::dec_refcnt(diff, utcb); }
00887 };
00888
00889 template< typename Derived, typename Base1, typename Base2,
00890 long PROTO, typename S_DEMAND >
00891 Type_info const *const
00892 Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>::__Kobject_typeid::b
00893 [] =
00894 {
00895 &Base1::__Kobject_typeid::_m,
00896 &Base2::__Kobject_typeid::_m
00897 };
00898
00900 template< typename Derived, typename Base1, typename Base2,
00901 long PROTO, typename S_DEMAND >
00902 L4_____GEN_TI(Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>
00903);
00904
00905
00906
00907
00927 template<
00928 typename Derived,
00929 typename Base1,
00930 typename Base2,
00931 typename Base3,
00932 long PROTO = PROTO_ANY,
00933 typename S_DEMAND = Type_info::Demand_t<>
00934 >
00935 struct Kobject_3t : Base1, Base2, Base3
00936 {
00937 protected:
00939 typedef Derived Class;
00941 typedef Typeid::Iface<PROTO, Derived> __Iface;
00943 typedef Typeid::Merge_list<
00944 Typeid::Iface_list<__Iface>,
00945 Typeid::Merge_list<
00946 typename Base1::__Iface_list,
00947 Typeid::Merge_list<
00948 typename Base2::__Iface_list,
00949 typename Base3::__Iface_list
00950 >
00951 >
00952 > __Iface_list;
00953
00955 static void __check_protocols__()
00956 {
00957 typedef typename Base1::__Iface_list Base1_proto_list;
00958 typedef typename Base2::__Iface_list Base2_proto_list;
00959 typedef typename Base3::__Iface_list Base3_proto_list;
00960
00961 typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00962 typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00963 typedef Typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;
00964
00965 static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00966 static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00967 static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");
00968
00969 typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Conflict_bases12;
00970 typedef Typeid::Conflict<Base1_proto_list, Base3_proto_list> Conflict_bases13;
00971 typedef Typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;
00972
00973 static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Base1 and Base2");
00974 static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Base1 and Base3");
00975 static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and Base3");
00976 }

```

```

00977
00978 // disambiguate cap()
00979 l4_cap_idx_t cap() const throw()
00980 { return Base1::cap(); }
00981
00982 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); }
00983
00984 L4_____GEN_TI_MEMBERS(Type_info::Demand_union_t<
00985 Type_info::Demand_union_t<
00986 typename Base1::__Kobject_typeid::Demand,
00987 typename Base2::__Kobject_typeid::Demand>,
00988 typename Base3::__Kobject_typeid::Demand>
00989)
00990
00991 public:
00992 // Provide non-ambiguous conversion to Kobject
00993 operator Kobject const & () const
00994 { return *static_cast<Base1 const *>(this); }
00995
00996 // Provide non-ambiguous access of dec_refcnt()
00997 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb =
00998 l4_utcb())
00999 { return Base1::dec_refcnt(diff, utcb); }
01000 };
01001
01002 template< typename Derived, typename Base1, typename Base2, typename Base3,
01003 long PROTO, typename S_DEMAND >
01004 Type_info const *const
01005 Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::_b
01006 [] =
01007 {
01008 &Base1::__Kobject_typeid::_m,
01009 &Base2::__Kobject_typeid::_m,
01010 &Base3::__Kobject_typeid::_m
01011 };
01012
01013 template< typename Derived, typename Base1, typename Base2, typename Base3,
01014 long PROTO, typename S_DEMAND >
01015 L4_____GEN_TI(Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>
01016);
01017
01018 #if __cplusplus >= 201103L
01019 namespace L4 {
01020
01021 template< typename ...T >
01022 struct Kobject_demand;
01023
01024 template<>
01025 struct Kobject_demand<> : Type_info::Demand_t<> {};
01026
01027 template<typename T>
01028 struct Kobject_demand<T> : Kobject_typeid<T>::Demand {};
01029
01030 template<typename T1, typename ...T2>
01031 struct Kobject_demand<T1, T2...> :
01032 Type_info::Demand_union_t<typename Kobject_typeid<T1>::Demand,
01033 Kobject_demand<T2...> >
01034 {};
01035
01036 namespace Typeid_xx {
01037
01038 template<typename ...LISTS>
01039 struct Merge_list;
01040
01041 template<typename L>
01042 struct Merge_list<L> : L {};
01043
01044 template<typename L1, typename L2>
01045 struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};
01046
01047 template<typename L1, typename L2, typename ...LISTS>
01048 struct Merge_list<L1, L2, LISTS...> :
01049 Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};
01050
01051 template< typename I, typename ...LIST >
01052 struct Iface_conflict;
01053
01054 template< typename I >
01055 struct Iface_conflict<I> : Typeid::False {};
01056
01057 template< typename I, typename L, typename ...LIST >
01058 struct Iface_conflict<I, L, LIST...> :
01059 Typeid::Bool<Typeid::Iface_conflict<typename I::type, typename L::type>::value
01060
```

```

01071 || Iface_conflict<I, LIST...>::value>
01072 {};
```

```

01073
01074 template< typename ...LIST >
01075 struct Conflict;
```

```

01076
01077 template< typename L >
01078 struct Conflict<L> : Typeid::False {};
```

```

01079
01080 template< typename L1, typename L2, typename ...LIST >
01081 struct Conflict<L1, L2, LIST...> :
01082 Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
01083 || Conflict<L1, LIST...>::value
01084 || Conflict<L2, LIST...>::value>
01085 {};
```

```

01086
01087 template< typename T >
01088 struct Is_demand
01089 {
01090 static long test(Type_info::Demand const *);
01091 static char test(...);
01092 enum { value = sizeof(test((T*)0)) == sizeof(long) };
01093 };
01094
01095 template< typename T, typename ... >
01096 struct First : T { typedef T type; };
01097 } // Typeid
01098
01104 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01105 struct __Kobject_base : BASES...
01106 {
01107 protected:
01108 typedef Derived Class;
01109 typedef Typeid::Iface<PROTO, Derived> __Iface;
01110 typedef Typeid_xx::Merge_list<
01111 Typeid::Iface_list<__Iface>,
01112 typename BASES::__Iface_list...
01113 > __Iface_list;
01114
01115 static void __check_protocols__()
01116 {
01117 typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
01118 static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");
01119
01120 typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
01121 static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
01122 }
01123
01124 // disambiguate cap()
01125 l4_cap_idx_t cap() const throw()
01126 { return Typeid_xx::First<BASES...>::type::cap(); }
01127
01128 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); }
01129
01130 L4__GEN_TI_MEMBERS(Kobject_demand<BASES...>)
01131
01132 private:
01133 // This function returns the first base class (used below)
01134 template<typename B1, typename ...> struct Basel { typedef B1 type; };
01135
01136 public:
01137 // Provide non-ambiguous conversion to Kobject
01138 operator Kobject const & () const
01139 { return *static_cast<typename Basel<BASES...>::type const *>(this); }
01140
01141 // Provide non-ambiguous access of dec_refcnt()
01142 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb =
01143 l4_utcb())
01144 { return Basel<BASES...>::type::dec_refcnt(diff, utcb); }
01145 };
01146
01147 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01148 Type_info const *const
01149 __Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::b[] =
01150 {
01151 (&BASES::__Kobject_typeid::_m)...
01152 };
01153
01154 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01155 L4__GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);
01156
01157 // Test if there is a Demand argument to Kobject_x
01158 template< typename Derived, long PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
01159 struct __Kobject_x_proto;
01160
01161 // YES: pass it to __Kobject_base
```



```

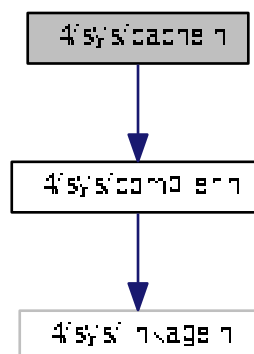
01162 template< typename Derived, long PROTO, typename DEMAND, typename ...BASES>
01163 struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
01164 __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};
01165
01166 // NO: pass it empty Type_info::Demand_t
01167 template< typename Derived, long PROTO, typename B1, typename ...BASES>
01168 struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
01169 __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};
01170
01178 template< long P = PROTO_EMPTY >
01179 struct Proto_t {};
01180
01194 template< typename Derived, typename ...ARGS >
01195 struct Kobject_x;
01196
01197 template< typename Derived, typename A, typename ...ARGS >
01198 struct Kobject_x<Derived, A, ARGS...> :
01199 __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01200 {};
01201
01202 template< typename Derived, long PROTO, typename A, typename ...ARGS >
01203 struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
01204 __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01205 {};
01206
01207 }
01208 #endif
01209
01210 #undef L4___GEN_TI
01211 #undef L4___GEN_TI_MEMBERS
01212

```

## 15.261 l4/sys/cache.h File Reference

Cache-consistency functions.

#include <l4/sys/compiler.h>  
 Include dependency graph for cache.h:



## Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
 Cache clean a range in D-cache.
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)

*Cache flush a range.*

- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)

*Cache invalidate a range.*

- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)

*Make memory coherent between I-cache and D-cache.*

- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)

*Make memory coherent for use with external memory.*

- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)

*Make memory coherent for use with external memory.*

### 15.261.1 Detailed Description

Cache-consistency functions.

#### Date

2007-11

#### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).

## 15.262 cache.h

```

00001
00011 /*
00012 * (c) 2007-2009 Author(s)
00013 * economic rights: Technische Universität Dresden (Germany)
00014 *
00015 * This file is part of TUD:OS and distributed under the terms of the
00016 * GNU General Public License 2.
00017 * Please see the COPYING-GPL-2 file for details.
00018 *
00019 * As a special exception, you may use this file as part of a free software
00020 * library without restriction. Specifically, if other files instantiate
00021 * templates or use macros or inline functions from this file, or you compile
00022 * this file and link it with other files to produce an executable, this
00023 * file does not by itself cause the resulting executable to be covered by
00024 * the GNU General Public License. This exception does not however
00025 * invalidate any other reasons why the executable file might be covered by
00026 * the GNU General Public License.
00027 */
00028
00029 #ifndef __L4SYS__INCLUDE__CACHE_H__
00030 #define __L4SYS__INCLUDE__CACHE_H__
00031
00032 #include <l4/sys/compiler.h>
00033
00042 EXTERN_C_BEGIN
00043
00058 L4_INLINE int
00059 l4_cache_clean_data(unsigned long start,
00060 unsigned long end) L4_NOTHROW;
00061
00076 L4_INLINE int
00077 l4_cache_flush_data(unsigned long start,
00078 unsigned long end) L4_NOTHROW;
00079
00098 L4_INLINE int
00099 l4_cache_inv_data(unsigned long start,
00100 unsigned long end) L4_NOTHROW;
00101

```

```

00113 L4_INLINE int
00114 l4_cache_coherent(unsigned long start,
00115 unsigned long end) L4_NOTHROW;
00116
00128 L4_INLINE int
00129 l4_cache_dma_coherent(unsigned long start,
00130 unsigned long end) L4_NOTHROW;
00131
00136 L4_INLINE int
00137 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00138
00139 EXTERN_C_END
00140
00141 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */

```

## 15.263 arm/l4/sys/cache.h File Reference

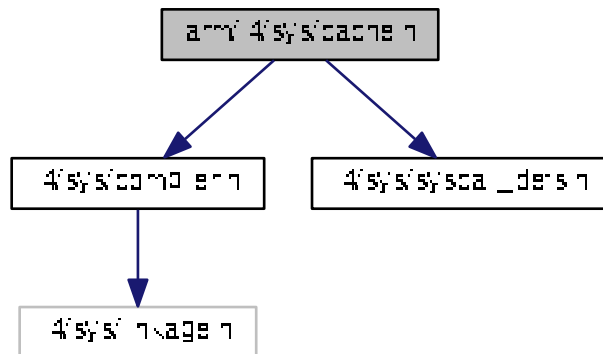
Cache functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>

```

Include dependency graph for cache.h:



## Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*

## 15.263.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).

## 15.264 cache.h

```

00001
00009 /*
00010 * (c) 2007-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 #include_next <l4/sys/cache.h>
00033
00037 L4_INLINE void
00038 l4_cache_op_arm_call(unsigned long op,
00039 unsigned long start,
00040 unsigned long end);
00041
00042 L4_INLINE void
00043 l4_cache_op_arm_call(unsigned long op,
00044 unsigned long start,
00045 unsigned long end)
00046 {
00047 register unsigned long _op __asm__ ("r0") = op;
00048 register unsigned long _start __asm__ ("r1") = start;
00049 register unsigned long _end __asm__ ("r2") = end;
00050
00051 __asm__ __volatile__
00052 ("@ l4_cache_op_arm_call(start) \n\t"
00053 "mov lr, pc \n\t"
00054 "mov pc, %[sc] \n\t"
00055 "@ l4_cache_op_arm_call(end) \n\t"
00056 :
00057 "=r" (_op),
00058 "=r" (_start),
00059 "=r" (_end)
00060 :
00061 [sc] "i" (L4_SYSCALL_MEM_OP),
00062 "0" (_op),
00063 "1" (_start),
00064 "2" (_end)
00065 :
00066 "cc", "memory", "lr"
00067);

```

```

00068 }
00069
00070 enum L4_mem_cache_ops
00071 {
00072 L4_MEM_CACHE_OP_CLEAN_DATA = 0,
00073 L4_MEM_CACHE_OP_FLUSH_DATA = 1,
00074 L4_MEM_CACHE_OP_INV_DATA = 2,
00075 L4_MEM_CACHE_OP_COHERENT = 3,
00076 L4_MEM_CACHE_OP_DMA_COHERENT = 4,
00077 L4_MEM_CACHE_OP_DMA_COHERENT_FULL = 5,
00078 L4_MEM_CACHE_OP_L2_CLEAN = 6,
00079 L4_MEM_CACHE_OP_L2_FLUSH = 7,
00080 L4_MEM_CACHE_OP_L2_INV = 8,
00081 };
00082
00083 L4_INLINE int
00084 l4_cache_clean_data(unsigned long start,
00085 unsigned long end) L4_NOTHROW
00086 {
00087 l4_cache_op_arm_call(L4_MEM_CACHE_OP_CLEAN_DATA, start, end);
00088 return 0;
00089 }
00090
00091 L4_INLINE int
00092 l4_cache_flush_data(unsigned long start,
00093 unsigned long end) L4_NOTHROW
00094 {
00095 l4_cache_op_arm_call(L4_MEM_CACHE_OP_FLUSH_DATA, start, end);
00096 return 0;
00097 }
00098
00099 L4_INLINE int
00100 l4_cache_inv_data(unsigned long start,
00101 unsigned long end) L4_NOTHROW
00102 {
00103 l4_cache_op_arm_call(L4_MEM_CACHE_OP_INV_DATA, start, end);
00104 return 0;
00105 }
00106
00107 L4_INLINE int
00108 l4_cache_coherent(unsigned long start,
00109 unsigned long end) L4_NOTHROW
00110 {
00111 l4_cache_op_arm_call(L4_MEM_CACHE_OP_COHERENT, start, end);
00112 return 0;
00113 }
00114
00115 L4_INLINE int
00116 l4_cache_dma_coherent(unsigned long start,
00117 unsigned long end) L4_NOTHROW
00118 {
00119 l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT, start, end);
00120 return 0;
00121 }
00122
00123 L4_INLINE int
00124 l4_cache_dma_coherent_full(void) L4_NOTHROW
00125 {
00126 l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT_FULL, 0, 0);
00127 return 0;
00128 }
00129
00130 L4_INLINE int
00131 l4_cache_l2_clean(unsigned long start,
00132 unsigned long end) L4_NOTHROW
00133 {
00134 l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_CLEAN, start, end);
00135 return 0;
00136 }
00137
00138 L4_INLINE int
00139 l4_cache_l2_flush(unsigned long start,
00140 unsigned long end) L4_NOTHROW
00141 {
00142 l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_FLUSH, start, end);
00143 return 0;
00144 }
00145
00146 L4_INLINE int
00147 l4_cache_l2_inv(unsigned long start,
00148 unsigned long end) L4_NOTHROW
00149 {
00150 l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_INV, start, end);
00151 return 0;
00152 }
00153
00154 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */

```

## 15.265 amd64/l4/sys/cache.h File Reference

Cache functions.

### Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*

### 15.265.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

## 15.266 cache.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029 unsigned long end) L4_NOTHROW
00030 {
00031 (void)start; (void)end;
00032 return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037 unsigned long end) L4_NOTHROW

```

```

00038 {
00039 (void)start; (void)end;
00040 return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045 unsigned long end) L4_NOTHROW
00046 {
00047 (void)start; (void)end;
00048 return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053 unsigned long end) L4_NOTHROW
00054 {
00055 (void)start; (void)end;
00056 return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
00061 unsigned long end) L4_NOTHROW
00062 {
00063 (void)start; (void)end;
00064 return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070 return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__ */

```

## 15.267 x86/I4/sys/cache.h File Reference

Cache functions.

### Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*

### 15.267.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

## 15.268 cache.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #ifndef __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029 unsigned long end) L4_NOTHROW
00030 {
00031 (void)start; (void)end;
00032 return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037 unsigned long end) L4_NOTHROW
00038 {
00039 (void)start; (void)end;
00040 return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045 unsigned long end) L4_NOTHROW
00046 {
00047 (void)start; (void)end;
00048 return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053 unsigned long end) L4_NOTHROW
00054 {
00055 (void)start; (void)end;
00056 return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
00061 unsigned long end) L4_NOTHROW
00062 {
00063 (void)start; (void)end;
00064 return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070 return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CACHE_H__ */

```

## 15.269 l4/sys/capability File Reference

[L4::Cap](#) related definitions.

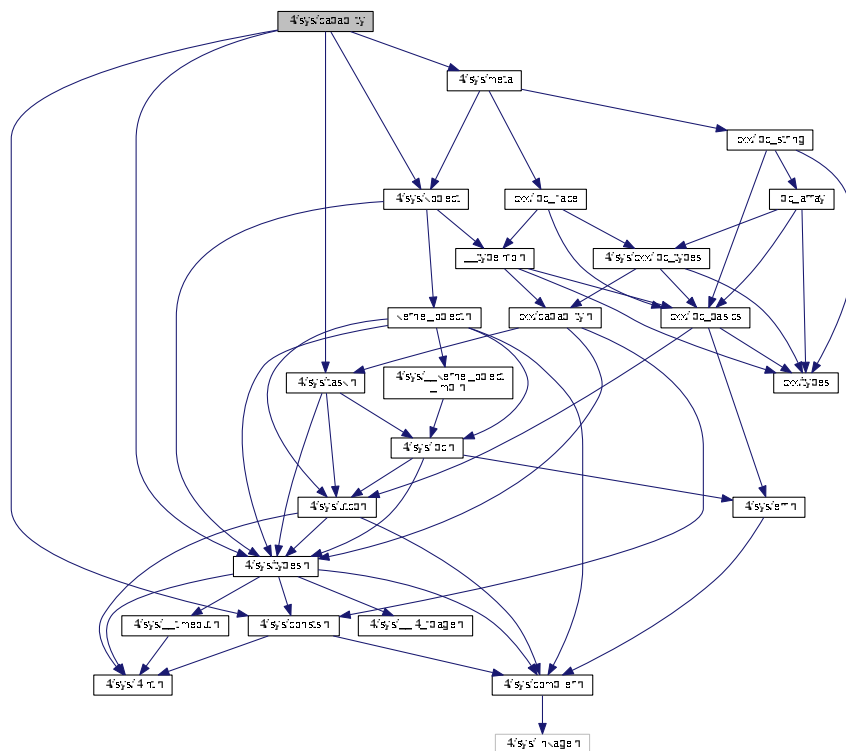
```

#include <l4/sys/consts.h>
#include <l4/sys/types.h>

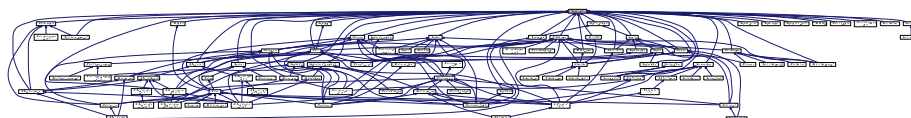
```



```
#include <l4/sys/kobject>
#include <l4/sys/task.h>
#include <l4/sys/meta>
Include dependency graph for capability:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- L4  
*L4 low-level kernel interface.*

## Macros

- `#define L4_DISABLE_COPY(_class)`  
*Disable copy of a class.*

## Functions

- `template<typename T, typename F >`  
`Cap< T > L4::cap_dynamic_cast (Cap< F > const &c) throw ()`  
*dynamic\_cast for capabilities.*

### 15.269.1 Detailed Description

[L4::Cap](#) related definitions.

#### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [capability](#).

### 15.269.2 Macro Definition Documentation

#### 15.269.2.1 L4\_DISABLE\_COPY

```
#define L4_DISABLE_COPY(
 _class)
```

#### Value:

```
public:
 _class(_class const &) = delete; \
 _class operator = (_class const &) = delete; \
private:
```

Disable copy of a class.

#### Parameters

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>_class</code> | Name of the class that shall not have value copy semantics. |
|---------------------|-------------------------------------------------------------|

The typical use of this is:

```
class Non_value
{
 L4_DISABLE_COPY(Non_value)

 ...
}
```

Definition at line 61 of file [capability](#).

## 15.270 capability

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010 * (c) 2008-2009,2015 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
```

```

00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYINGING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/consts.h>
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/kobject>
00031 #include <l4/sys/task.h>
00032
00033 namespace L4
00034 {
00035
00036 /* Forward declarations for our kernel object classes. */
00037 class Task;
00038 class Thread;
00039 class Factory;
00040 class Irq;
00041 class Log;
00042 class Vm;
00043 class Kobject;
00044
00060 #if __cplusplus >= 201103L
00061 # define L4_DISABLE_COPY(_class) \
00062 public: \
00063 _class(_class const &) = delete; \
00064 _class operator = (_class const &) = delete; \
00065 private:
00066 #else
00067 # define L4_DISABLE_COPY(_class) \
00068 private: \
00069 _class(_class const &); \
00070 _class operator = (_class const &);
00071 #endif
00072
00073
00074 #define L4_KOBJECT_DISABLE_COPY(_class) \
00075 protected: \
00076 _class(); \
00077 L4_DISABLE_COPY(_class)
00078
00079
00080 #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00081
00082 inline l4_msgtag_t
00083 Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const throw()
00084 {
00085 return is_valid() ? l4_task_cap_valid_u(task.cap(), _c, u)
00086 : l4_msgtag(0, 0, 0, 0);
00087 }
00088
00089 inline l4_msgtag_t
00090 Cap_base::validate(l4_utcb_t *u) const throw()
00091 {
00092 return is_valid() ? l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u)
00093 : l4_msgtag(0, 0, 0, 0);
00094 }
00095
00096 }; // namespace L4
00097
00098 #include <l4/sys/meta>
00099
00100 namespace L4 {
00101
00123 template< typename T, typename F >
00124 inline
00125 Cap<T> cap_dynamic_cast(Cap<F> const &c) throw()
00126 {
00127 if (!c.is_valid())
00128 return Cap<T>::Invalid;
00129
00130 Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
00131 Type_info const *m = kobject_typeid<T>();
00132 if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
00133 return Cap<T>(c.cap());
00134
00135 // FIXME: use generic checker

```

```

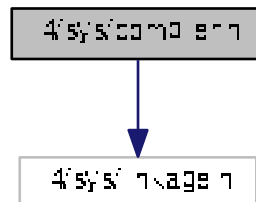
00136 #if 0
00137 if (l4_error(mc->supports(T::kobject_proto())) > 0)
00138 return Cap<T>(c.cap());
00139 #endif
00140
00141 return Cap<T>::Invalid;
00142 }
00143
00144 }
```

## 15.271 l4/sys/compiler.h File Reference

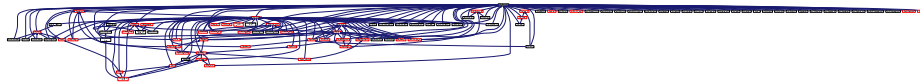
L4 compiler related defines.

```
#include <l4/sys/linkage.h>
```

Include dependency graph for compiler.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define L4_ALWAYS_INLINE`  
L4 Inline function attribute.
- `#define L4_NOTHROW`  
Mark a function declaration and definition as never throwing an exception.
- `#define EXTERN_C_BEGIN`  
Start section with C types and functions.
- `#define EXTERN_C_END`  
End section with C types and functions.
- `#define EXTERN_C`  
Mark C types and functions.
- `#define __END_DECLS`  
End section with C types and functions.
- `#define L4_NORETURN`

- *Noreturn function attribute.*
- `#define L4_NOINSTRUMENT`  
*No instrumentation function attribute.*
- `#define L4_HIDDEN`  
*Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.*
- `#define L4_LIKELY(x)`  
*Expression is likely to execute.*
- `#define L4_UNLIKELY(x)`  
*Expression is unlikely to execute.*
- `#define L4_STICKY(x)`  
*Mark symbol sticky (even not there)*
- `#define L4_DEPRECATED(s)`  
*Mark symbol deprecated.*
- `#define L4_stringify_helper(x)`  
*stringify helper.*
- `#define L4_stringify(x)`  
*stringify.*

## Functions

- `void l4_barrier (void)`  
*Memory barrier.*
- `void l4_mb (void)`  
*Memory barrier.*
- `void l4_wmb (void)`  
*Write memory barrier.*

### 15.271.1 Detailed Description

[L4](#) compiler related defines.

Definition in file [compiler.h](#).

## 15.272 compiler.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00011 * Jork Löser <jork@os.inf.tu-dresden.de>,
00012 * Ronald Aigner <ra3@os.inf.tu-dresden.de>
00013 * economic rights: Technische Universität Dresden (Germany)
00014 *
00015 * This file is part of TUD:OS and distributed under the terms of the
00016 * GNU General Public License 2.
00017 * Please see the COPYING-GPL-2 file for details.
00018 *
00019 * As a special exception, you may use this file as part of a free software
00020 * library without restriction. Specifically, if other files instantiate
00021 * templates or use macros or inline functions from this file, or you compile
00022 * this file and link it with other files to produce an executable, this
00023 * file does not by itself cause the resulting executable to be covered by
00024 * the GNU General Public License. This exception does not however
00025 * invalidate any other reasons why the executable file might be covered by
00026 * the GNU General Public License.

```

```

00027 */
00028 /*****
00029 #ifndef __L4_COMPILER_H__
00030 #define __L4_COMPILER_H__
00031
00032 #if !defined(__ASSEMBLY__) && !defined(__ASSEMBLER__)
00033
00040
00045 #ifndef L4_INLINE
00046 #ifndef __cplusplus
00047 # ifdef __OPTIMIZE__
00048 # define L4_INLINE_STATIC static __inline__
00049 # define L4_INLINE_EXTERN extern __inline__
00050 # ifdef __GNUC_STDC_INLINE__
00051 # define L4_INLINE L4_INLINE_STATIC
00052 # else
00053 # define L4_INLINE L4_INLINE_EXTERN
00054 # endif
00055 # else /* ! __OPTIMIZE__ */
00056 # define L4_INLINE static
00057 # endif /* ! __OPTIMIZE__ */
00058 #else /* __cplusplus */
00059 # define L4_INLINE inline
00060 #endif /* __cplusplus */
00061 #endif /* L4_INLINE */
00062
00067 #define L4_ALWAYS_INLINE L4_INLINE __attribute__((__always_inline__))
00068
00069
00070 #define L4_DECLARE_CONSTRUCTOR(func, prio) \
00071 static inline __attribute__((constructor(prio))) void func ## _ctor_func(void) { func(); }
00072
00073
00171 #ifndef __cplusplus
00172 # define L4_NOTHROW_A __attribute__((nothrow))
00173 # define L4_NOTHROW
00174 # define EXTERN_C_BEGIN
00175 # define EXTERN_C_END
00176 # define EXTERN_C
00177 # ifdef __BEGIN_DECLS
00178 # define __BEGIN_DECLS
00179 # endif
00180 # ifdef __END_DECLS
00181 # define __END_DECLS
00182 # endif
00183 # define L4_DEFAULT_PARAM(x)
00184 #else /* __cplusplus */
00185 # define L4_NOTHROW throw()
00186 # define EXTERN_C_BEGIN extern "C" {
00187 # define EXTERN_C_END }
00188 # define EXTERN_C extern "C"
00189 # ifdef __BEGIN_DECLS
00190 # define __BEGIN_DECLS extern "C" {
00191 # endif
00192 # ifdef __END_DECLS
00193 # define __END_DECLS }
00194 # endif
00195 # define L4_DEFAULT_PARAM(x) = x
00196 #endif /* __cplusplus */
00197
00202 #define L4_NORETURN __attribute__((noreturn))
00203
00204 #define L4_PURE __attribute__((pure))
00205
00210 #define L4_NOINSTRUMENT __attribute__((no_instrument_function))
00211 #ifndef L4_HIDDEN
00212 # define L4_HIDDEN __attribute__((visibility("hidden")))
00213 #endif
00214 #ifndef L4_EXPORT
00215 # define L4_EXPORT __attribute__((visibility("default")))
00216 #endif
00217 #ifndef L4_EXPORT_TYPE
00218 # ifdef __cplusplus
00219 # define L4_EXPORT_TYPE __attribute__((visibility("default")))
00220 # else
00221 # define L4_EXPORT_TYPE
00222 # endif
00223 #endif
00224 #define L4_STRONG_ALIAS(name, aliasname) L4__STRONG_ALIAS(name, aliasname)
00225 #define L4__STRONG_ALIAS(name, aliasname) \
00226 extern __typeof (name) aliasname __attribute__((alias (#name)));
00227
00228
00229 #endif /* !__ASSEMBLY__ */
00230
00231 #include <l4/sys/linkage.h>
00232

```

```

00233 #define L4_LIKELY(x) __builtin_expect((x),1)
00234 #define L4_UNLIKELY(x) __builtin_expect((x),0)
00235
00236 /* Make sure that the function is not removed by optimization. Without the
00237 * "used" attribute, unreferenced static functions are removed. */
00238 #define L4_STICKY(x) __attribute__((used)) x
00239 #define L4_DEPRECATED(s) __attribute__((deprecated(s)))
00240
00241 #ifndef __GXX_EXPERIMENTAL_CXX0X__
00242 #ifndef static_assert
00243 #define static_assert(x, y) \
00244 do { (void)sizeof(char[!(x)]); } while (0)
00245 #endif
00246 #endif
00247
00248 #define L4_stringify_helper(x) #x
00249 #define L4_stringify(x) L4_stringify_helper(x)
00250
00251 #ifndef __ASSEMBLER__
00252
00255 L4_INLINE void l4_barrier(void);
00256
00260 L4_INLINE void l4_mb(void);
00261
00265 L4_INLINE void l4_wmb(void);
00266
00267
00268 /* Implementations */
00269 L4_INLINE void l4_barrier(void)
00270 {
00271 __asm__ __volatile__ (" : : : \"memory\"");
00272 }
00273
00274 L4_INLINE void l4_mb(void)
00275 {
00276 __asm__ __volatile__ (" : : : \"memory\"");
00277 }
00278
00279 L4_INLINE void l4_wmb(void)
00280 {
00281 __asm__ __volatile__ (" : : : \"memory\"");
00282 }
00283 #endif
00284
00287 #endif /* !__L4_COMPILER_H__ */

```

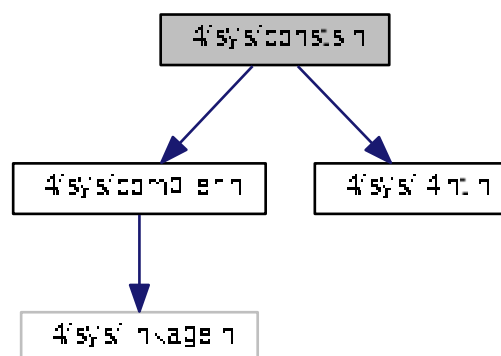
## 15.273 l4/sys/consts.h File Reference

Common constants.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define L4_PAGESIZE`  
*Minimal page size (in bytes).*
- `#define L4_PAGEMASK`  
*Mask for the page number.*
- `#define L4_LOG2_PAGESIZE`  
*Number of bits used for page offset.*
- `#define L4_SUPERPAGESIZE`  
*Size of a large page.*
- `#define L4_SUPERPAGEMASK`  
*Mask for the number of a large page.*
- `#define L4_LOG2_SUPERPAGESIZE`  
*Number of bits used as offset for a large page.*
- `#define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)`  
*Invalid address as pointer type.*

## Enumerations

- `enum l4_syscall_flags_t {`  
`L4_SYSF_NONE, L4_SYSF_SEND, L4_SYSF_RECV, L4_SYSF_OPEN_WAIT,`  
`L4_SYSF_REPLY, L4_SYSF_CALL, L4_SYSF_WAIT, L4_SYSF_SEND_AND_WAIT,`  
`L4_SYSF_REPLY_AND_WAIT }`  
*Capability selector flags.*
- `enum l4_cap_consts_t { L4_CAP_SHIFT, L4_CAP_SIZE, L4_CAP_MASK, L4_INVALID_CAP }`  
*Constants related to capability selectors.*
- `enum l4_unmap_flags_t { L4_FP_ALL_SPACES, L4_FP_DELETE_OBJ, L4_FP_OTHER_SPACES }`  
*Flags for the unmap operation.*
- `enum l4_msg_item_consts_t {`  
`L4_ITEM_MAP = 8, L4_ITEM_CONT = 1, L4_MAP_ITEM_GRANT = 2, L4_MAP_ITEM_MAP = 0,`  
`L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2, L4_RCV_ITEM_LOCAL_ID = 4 }`  
*Constants for message items.*
- `enum l4_buffer_desc_consts_t { L4_BDR_MEM_SHIFT = 0, L4_BDR_IO_SHIFT = 5, L4_BDR_OBJ_SHIFT = 10 }`  
*Constants for buffer descriptors.*
- `enum l4_default_caps_t {`  
`L4_BASE_TASK_CAP, L4_BASE_FACTORY_CAP, L4_BASE_THREAD_CAP, L4_BASE_PAGER_CAP,`  
`L4_BASE_LOG_CAP, L4_BASE_ICU_CAP, L4_BASE_SCHEDULER_CAP, L4_BASE_IOMMU_CAP,`  
`L4_BASE_DEBUGGER_CAP, L4_BASE_ARM_SMCCC_CAP, L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST - 1 }`  
*Default capabilities setup for the initial tasks.*
- `enum l4_addr_consts_t { L4_INVALID_ADDR = ~0UL }`  
*Address related constants.*



## Functions

- [l4\\_addr\\_t l4\\_trunc\\_page \(l4\\_addr\\_t address\)](#) [L4\\_NOTHROW](#)  
*Round an address down to the next lower page boundary.*
- [l4\\_addr\\_t l4\\_trunc\\_size \(l4\\_addr\\_t address, unsigned char bits\)](#) [L4\\_NOTHROW](#)  
*Round an address down to the next lower flex page with size bits.*
- [l4\\_addr\\_t l4\\_round\\_page \(l4\\_addr\\_t address\)](#) [L4\\_NOTHROW](#)  
*Round address up to the next page.*
- [l4\\_addr\\_t l4\\_round\\_size \(l4\\_umword\\_t value, unsigned char bits\)](#) [L4\\_NOTHROW](#)  
*Round value up to the next alignment with bits size.*

### 15.273.1 Detailed Description

Common constants.

Definition in file [consts.h](#).

## 15.274 consts.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4_SYS__INCLUDE__CONSTS_H__
00027 #define __L4_SYS__INCLUDE__CONSTS_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031
00039 enum l4_syscall_flags_t
00040 {
00048 L4_SYSF_NONE = 0x00,
00049
00058 L4_SYSF_SEND = 0x01,
00059
00069 L4_SYSF_RECV = 0x02,
00070
00080 L4_SYSF_OPEN_WAIT = 0x04,
00081
00089 L4_SYSF_REPLY = 0x08,
00090
00097 L4_SYSF_CALL = L4_SYSF_SEND | L4_SYSF_RECV,
00098
00105 L4_SYSF_WAIT = L4_SYSF_OPEN_WAIT |
00106 L4_SYSF_RECV,
00113 L4_SYSF_SEND_AND_WAIT = L4_SYSF_OPEN_WAIT |
00114 L4_SYSF_CALL,
00121 L4_SYSF_REPLY_AND_WAIT = L4_SYSF_WAIT |
00122 L4_SYSF_SEND | L4_SYSF_REPLY
00123 };

```

```

00128 enum l4_cap_consts_t
00129 {
00131 L4_CAP_SHIFT = 12UL,
00133 L4_CAP_SIZE = 1UL << L4_CAP_SHIFT,
00134 L4_CAP_OFFSET = 1UL << L4_CAP_SHIFT,
00139 L4_CAP_MASK = ~0UL << (L4_CAP_SHIFT - 1),
00141 L4_INVALID_CAP = ~0UL << (L4_CAP_SHIFT - 1),
00142
00143 L4_INVALID_CAP_BIT = 1UL << (L4_CAP_SHIFT - 1),
00144 };
00145
00146 enum l4_sched_consts_t
00147 {
00148 L4_SCHED_MIN_PRIO = 0,
00149 L4_SCHED_MAX_PRIO = 255,
00150 };
00151
00157 enum l4_unmap_flags_t
00158 {
00165 L4_FP_ALL_SPACES = 0x80000000UL,
00166
00173 L4_FP_DELETE_OBJ = 0xc0000000UL,
00174
00180 L4_FP_OTHER_SPACES = 0x0UL
00181 };
00182
00187 enum l4_msg_item_consts_t
00188 {
00189 L4_ITEM_MAP = 8,
00190
00195 L4_ITEM_CONT = 1,
00196
00197 // send
00198 L4_MAP_ITEM_GRANT = 2,
00199 L4_MAP_ITEM_MAP = 0,
00200
00201 // receive
00206 L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2,
00207
00212 L4_RCV_ITEM_LOCAL_ID = 4,
00213 };
00214
00219 enum l4_buffer_desc_consts_t
00220 {
00221 L4_BDR_MEM_SHIFT = 0,
00222 L4_BDR_IO_SHIFT = 5,
00223 L4_BDR_OBJ_SHIFT = 10,
00224 L4_BDR_OFFSET_MASK = (1UL << 20) - 1,
00225 };
00226
00240 enum l4_default_caps_t
00241 {
00243 L4_BASE_TASK_CAP = 1UL << L4_CAP_SHIFT,
00245 L4_BASE_FACTORY_CAP = 2UL << L4_CAP_SHIFT,
00247 L4_BASE_THREAD_CAP = 3UL << L4_CAP_SHIFT,
00255 L4_BASE_PAGER_CAP = 4UL << L4_CAP_SHIFT,
00263 L4_BASE_LOG_CAP = 5UL << L4_CAP_SHIFT,
00265 L4_BASE_ICU_CAP = 6UL << L4_CAP_SHIFT,
00267 L4_BASE_SCHEDULER_CAP = 7UL << L4_CAP_SHIFT,
00274 L4_BASE_IOMMU_CAP = 8UL << L4_CAP_SHIFT,
00282 L4_BASE_DEBUGGER_CAP = 10UL << L4_CAP_SHIFT,
00289 L4_BASE_ARM_SMCCC_CAP = 11UL << L4_CAP_SHIFT,
00290
00292 L4_BASE_CAPS_LAST_P1,
00294 L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1
00295 };
00296
00307 #define L4_PAGESIZE (1UL << L4_PAGESHIFT)
00308
00316 #define L4_PAGEMASK (~(L4_PAGESIZE - 1))
00317
00325 #define L4_LOG2_PAGESIZE L4_PAGESHIFT
00326
00334 #define L4_SUPERPAGESIZE (1UL << L4_SUPERPAGESHIFT)
00335
00343 #define L4_SUPERPAGEMASK (~(L4_SUPERPAGESIZE - 1))
00344
00351 #define L4_LOG2_SUPERPAGESIZE L4_SUPERPAGESHIFT
00352
00363 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address)
00364 L4_NOTHROW;
00364 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW
00365 { return address & L4_PAGEMASK; }
00366
00374 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits)
00375 L4_NOTHROW;
00375 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits)

```

```

 L4_NOTHROW
00376 { return address & (~0UL << bits); }
00377
00388 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address)
 L4_NOTHROW;
00389 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
00390 { return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }
00391
00399 L4_INLINE l4_addr_t l4_round_size(l4_umword_t value, unsigned char bits)
 L4_NOTHROW;
00400 L4_INLINE l4_addr_t l4_round_size(l4_umword_t value, unsigned char bits)
 L4_NOTHROW
00401 { return (value + (1UL << bits) - 1) & (~0UL << bits); }
00402
00407 enum l4_addr_consts_t {
00409 L4_INVALID_ADDR = ~0UL
00410 };
00411
00416 #define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)
00417
00418 #ifndef NULL
00419 #ifndef __cplusplus
00420 # define NULL ((void *)0)
00424 #else
00425 # define NULL 0
00426 #endif
00427 #endif
00428
00429 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */

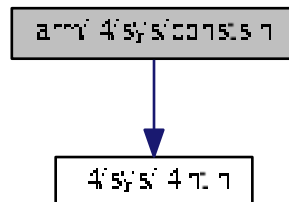
```

## 15.275 arm/l4/sys/consts.h File Reference

Common [L4](#) constants, arm version.

#include <l4/sys/l4int.h>

Include dependency graph for consts.h:



### Macros

- #define [L4\\_PAGESHIFT](#) 12  
*Size of a page, log2-based.*
- #define [L4\\_SUPERPAGESHIFT](#) 21  
*Size of a large page, log2-based.*

### 15.275.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).

## 15.276 consts.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_SYS_CONSTS_H
00026 #define __L4_SYS_CONSTS_H
00027
00028 /* L4 includes */
00029 #include <l4/sys/l4int.h>
00037 #define L4_PAGESHIFT 12
00038
00042 #define L4_SUPERPAGESHIFT 21
00043
00046 #include_next <l4/sys/consts.h>
00047
00048 #endif /* !__L4_SYS_CONSTS_H */

```

## 15.277 amd64/l4/sys/consts.h File Reference

Common [L4](#) constants, amd64 version.

### Macros

- `#define L4\_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4\_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*

### 15.277.1 Detailed Description

Common [L4](#) constants, amd64 version.

Definition in file [consts.h](#).

## 15.278 consts.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 /*****
00028 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00030
00035 #define L4_PAGESHIFT 12
00036
00041 #define L4_SUPERPAGESHIFT 21
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__ */

```

## 15.279 x86/l4/sys/consts.h File Reference

Common [L4](#) constants, x86 version.

### Macros

- `#define L4_PAGESHIFT 12`  
*Size of a page log2-based.*
- `#define L4_SUPERPAGESHIFT 22`  
*Size of a large page log2-based.*

### 15.279.1 Detailed Description

Common [L4](#) constants, x86 version.

Definition in file [consts.h](#).

## 15.280 consts.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 * Lars Reuther <reuther@os.inf.tu-dresden.de>
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 /*****
00028 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00030
00035 #define L4_PAGESHIFT 12
00036
00041 #define L4_SUPERPAGESHIFT 22
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */

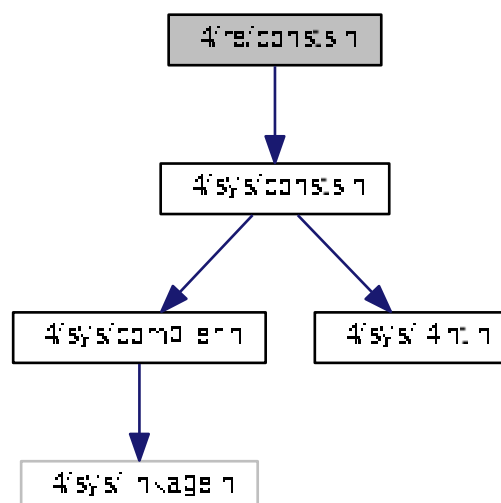
```

## 15.281 l4/re/consts.h File Reference

Constants.

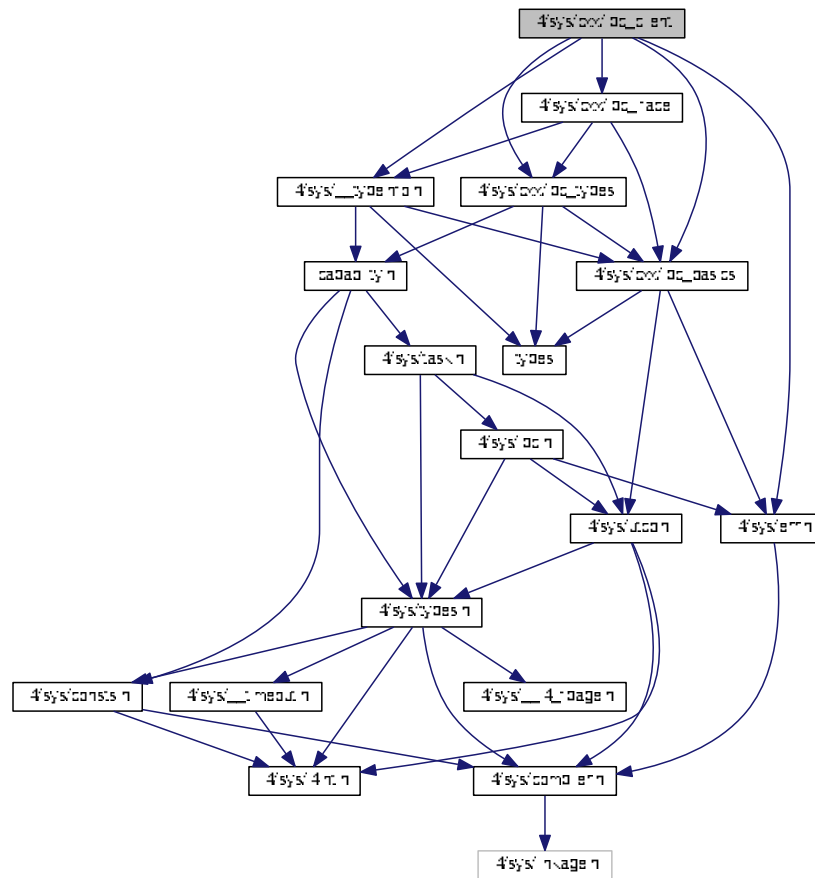
```
#include <l4/sys/consts.h>
```

Include dependency graph for consts.h:

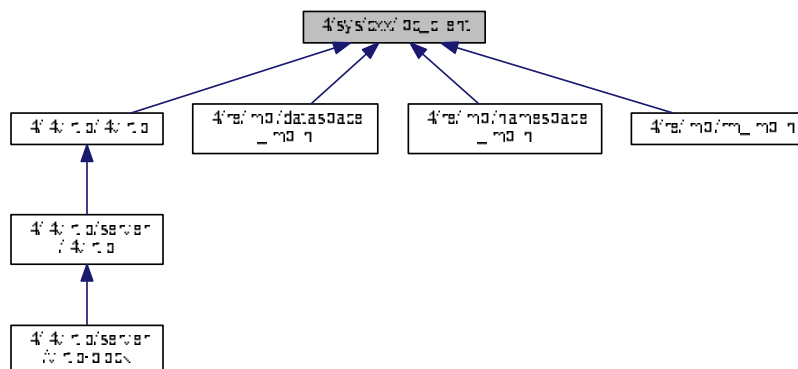




```
#include <linux/sys/err.h>
```



This graph shows which files directly or indirectly include this file:





## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*
- [L4::ipc](#)  
*IPC related functionality.*
- [L4::ipc::Msg](#)  
*IPC Message related functionality.*

## Macros

- `#define L4_RPC_DEF(name)`  
*Generate the definition of an RPC stub.*

### 15.283.1 Macro Definition Documentation

#### 15.283.1.1 L4\_RPC\_DEF

```
#define L4_RPC_DEF(
 name)
```

#### Value:

```
template struct L4::Ipcc::Msg::Rpc_call \
 <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
```

Generate the definition of an RPC stub.

#### Parameters

|             |                                                                                            |
|-------------|--------------------------------------------------------------------------------------------|
| <i>name</i> | The fully qualified method name to be implemented, this means <code>class::method</code> . |
|-------------|--------------------------------------------------------------------------------------------|

This macro generates the definition (implementation) for the given RPC interface method.

Definition at line 43 of file [ipc\\_client](#).

## 15.284 ipc\_client

```
00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
```

```

00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/__typeinfo.h>
00025 #include <l4/sys/err.h>
00026
00031 namespace L4 { namespace Ipc { namespace Msg {
00032 //-----
00033
00043 #define L4_RPC_DEF(name) \
00044 template struct L4::Ipc::Msg::Rpc_call \
00045 <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
00046
00047
00049 //-----
00050 //Implementation of the RPC call
00051 template<typename OP, typename C, typename FLAGS, typename R, typename ...ARGS>
00052 R L4_EXPORT
00053 Rpc_call<OP, C, R (ARGS...), FLAGS>::
00054 call(L4::Cap<C> cap, typename _Elem<ARGS>::arg_type ...a, l4_utcb_t *utcb) throw()
00055 {
00056 return Rpc_inline_call<OP, C, R (ARGS...), FLAGS>::call(cap, a..., utcb);
00057 }
00059
00060 } // namespace Msg
00061 } // namespace Ipc
00062 } // namespace L4
00063

```

## 15.285 l4/sys/cxx/ipc\_iface File Reference

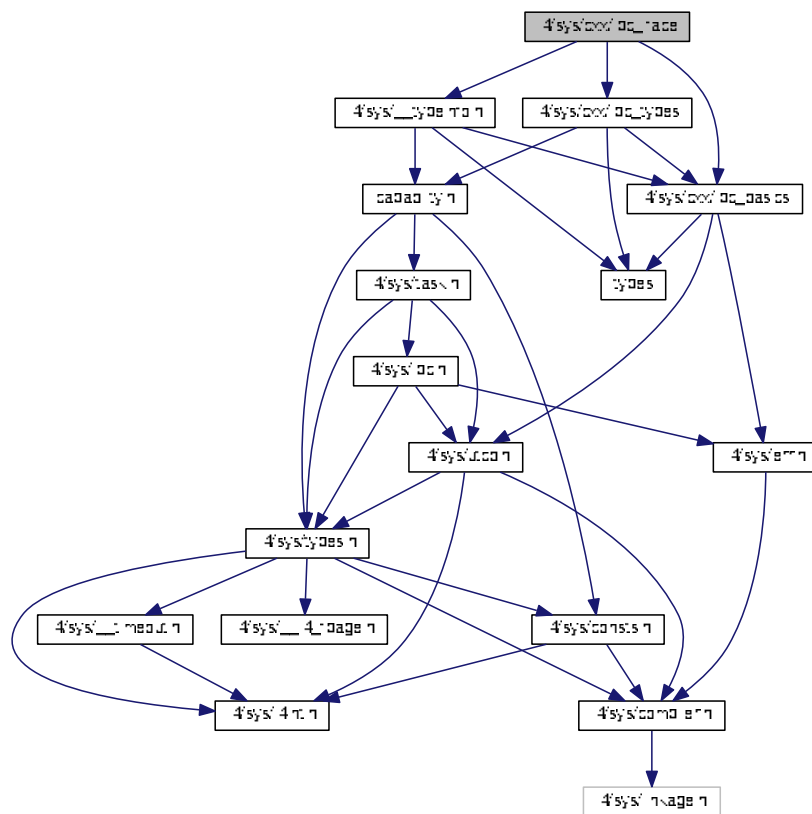
Interface Definition Language.

```

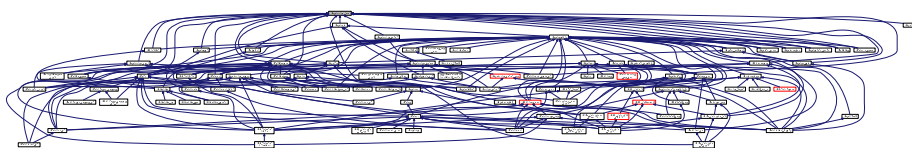
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>

```

Include dependency graph for ipc\_iface:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `L4::ipc::Call`  
*RPC attribute for a standard RPC call.*
- struct `L4::ipc::Call_zero_send_timeout`  
*RPC attribute for an RPC call, with zero send timeout.*
- struct `L4::ipc::Call_t< RIGHTS >`  
*RPC attribute for an RPC call with required rights.*
- struct `L4::ipc::Send_only`  
*RPC attribute for a send-only RPC.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*
- [L4::ipc](#)  
*IPC related functionality.*
- [L4::ipc::Msg](#)  
*IPC Message related functionality.*

## Macros

- `#define L4_INLINE_RPC_NF(res, name, args...)`  
*Define an inline RPC call type (the type only, no callable).*
- `#define L4_INLINE_RPC_NF_OP(op, res, name, args...)`  
*Define an inline RPC call type with specific opcode (the type only, no callable).*
- `#define L4_INLINE_RPC(res, name, args, attr...) res name args`  
*Define an inline RPC call (type and callable).*
- `#define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args`  
*Define an inline RPC call with specific opcode (type and callable).*
- `#define L4_RPC_NF(res, name, args...)`  
*Define an RPC call type (the type only, no callable).*
- `#define L4_RPC_NF_OP(op, res, name, args...)`  
*Define an RPC call type with specific opcode (the type only, no callable).*
- `#define L4_RPC(res, name, args, attr...) res name args`  
*Define an RPC call (type and callable).*
- `#define L4_RPC_OP(op, res, name, args, attr...) res name args`  
*Define an RPC call with specific opcode (type and callable).*

### 15.285.1 Detailed Description

Interface Definition Language.

See also

[L4\\_RPC](#), [L4\\_INLINE\\_RPC](#), [L4::ipc::Call](#) [L4::ipc::Send\\_only](#), [L4::ipc::Msg::Rpc\\_call](#), [L4::ipc::Msg::Rpc\\_↔  
inline\\_call](#)

Definition in file [ipc\\_iface](#).

### 15.285.2 Macro Definition Documentation

#### 15.285.2.1 L4\_INLINE\_RPC

```
#define L4_INLINE_RPC(
 res,
 name,
 args,
 attr...) res name args
```

Define an inline RPC call (type and callable).

## Parameters

|             |                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------|
| <i>res</i>  | The result type of the RPC call                                                                  |
| <i>name</i> | The name of the function ( <i>name_t</i> is used for the type.)                                  |
| <i>args</i> | The argument list of the RPC function.                                                           |
| <i>attr</i> | Optional RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Definition at line 458 of file [ipc\\_iface](#).

Referenced by [L4Re::Video::Goos::delete\\_view\(\)](#).

## 15.285.2.2 L4\_INLINE\_RPC\_NF

```
#define L4_INLINE_RPC_NF(
 res,
 name,
 args...)
```

## Value:

```
struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
 typedef L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> type; \
 L4_INLINE_RPC_SRV_FORWARD(name); \
}
```

Define an inline RPC call type (the type only, no callable).

## Parameters

|             |                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>res</i>  | The result type of the RPC call                                                                                                    |
| <i>name</i> | The name of the function ( <i>name_t</i> is used for the type.)                                                                    |
| <i>args</i> | The argument list of the RPC function, and RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line 429 of file [ipc\\_iface](#).

Referenced by [L4::Factory::create\(\)](#), [L4Re::Video::Goos::create\\_view\(\)](#), and [L4Re::Inhibitor::next\\_lock\\_info\(\)](#).

## 15.285.2.3 L4\_INLINE\_RPC\_NF\_OP

```
#define L4_INLINE_RPC_NF_OP(
 op,
```

```

 res,
 name,
 args...)

```

**Value:**

```

struct name##_t : L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args> \
{
 typedef L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args> type; \
 enum { Opcode = (op) }; \
 L4_INLINE_RPC_SRV_FORWARD(name); \
}

```

Define an inline RPC call type with specific opcode (the type only, no callable).

**Parameters**

|             |                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>op</i>   | The opcode number for this function                                                                                                |
| <i>res</i>  | The result type of the RPC call                                                                                                    |
| <i>name</i> | The name of the function ( <i>name_t</i> is used for the type.)                                                                    |
| <i>args</i> | The argument list of the RPC function, and RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line [442](#) of file [ipc\\_iface](#).

**15.285.2.4 L4\_INLINE\_RPC\_OP**

```

#define L4_INLINE_RPC_OP(
 op,
 res,
 name,
 args,
 attr...) res name args

```

Define an inline RPC call with specific opcode (type and callable).

**Parameters**

|             |                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------|
| <i>op</i>   | The opcode number for this function                                                              |
| <i>res</i>  | The result type of the RPC call                                                                  |
| <i>name</i> | The name of the function ( <i>name_t</i> is used for the type.)                                  |
| <i>args</i> | The argument list of the RPC function.                                                           |
| <i>attr</i> | Optional RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Definition at line [473](#) of file [ipc\\_iface](#).

Referenced by [L4::Scheduler::info\(\)](#), and [L4::lcu::info\(\)](#).

## 15.285.2.5 L4\_RPC

```
#define L4_RPC(
 res,
 name,
 args,
 attr...) res name args
```

Define an RPC call (type and callable).

## Parameters

|             |                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------|
| <i>res</i>  | The result type of the RPC call                                                                  |
| <i>name</i> | The name of the function ( <code>name_t</code> is used for the type.)                            |
| <i>args</i> | The argument list of the RPC function.                                                           |
| <i>attr</i> | Optional RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Definition at line 517 of file [ipc\\_iface](#).

Referenced by [L4Re::Rm::find\(\)](#), and [L4Re::Rm::reserve\\_area\(\)](#).

## 15.285.2.6 L4\_RPC\_NF

```
#define L4_RPC_NF(
 res,
 name,
 args...)
```

## Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
 typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;
 L4_INLINE_RPC_SRV_FORWARD(name);
}
```

```
\\
\\
\\
```

Define an RPC call type (the type only, no callable).

## Parameters

|             |                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>res</i>  | The result type of the RPC call                                                                                                    |
| <i>name</i> | The name of the function ( <code>name_t</code> is used for the type.)                                                              |
| <i>args</i> | The argument list of the RPC function, and RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Definition at line 486 of file [ipc\\_iface](#).

Referenced by [L4Re::Rm::find\(\)](#), and [L4Re::Rm::reserve\\_area\(\)](#).

## 15.285.2.7 L4\_RPC\_NF\_OP

```
#define L4_RPC_NF_OP (
 op,
 res,
 name,
 args...)
```

**Value:**

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
 typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;
 enum { Opcode = (op) };
 L4_INLINE_RPC_SRV_FORWARD(name);
}
```

```
\
/\
/\
/\
```

Define an RPC call type with specific opcode (the type only, no callable).

**Parameters**

|             |                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>op</i>   | The opcode number for this function                                                                                                |
| <i>res</i>  | The result type of the RPC call                                                                                                    |
| <i>name</i> | The name of the function ( <i>name_t</i> is used for the type.)                                                                    |
| <i>args</i> | The argument list of the RPC function, and RPC attributes ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.). |

Definition at line 501 of file [ipc\\_iface](#).

Referenced by [L4::Icu::bind\(\)](#), [L4::Icu::info\(\)](#), [L4::Icu::mask\(\)](#), [L4Re::Namespace::register\\_obj\(\)](#), [L4::Icu::set\\_mode\(\)](#), and [L4::Icu::unbind\(\)](#).

## 15.285.2.8 L4\_RPC\_OP

```
#define L4_RPC_OP (
 op,
 res,
 name,
 args,
 attr...) res name args
```

Define an RPC call with specific opcode (type and callable).

**Parameters**

|             |                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------|
| <i>op</i>   | The opcode number for this function                                                              |
| <i>res</i>  | The result type of the RPC call                                                                  |
| <i>name</i> | The name of the function ( <i>name_t</i> is used for the type.)                                  |
| <i>args</i> | The argument list of the RPC function.                                                           |
| <i>attr</i> | Optional RPC attributes ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.). |



Definition at line 532 of file [ipc\\_iface](#).

## 15.286 ipc\_iface

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/__typeinfo.h>
00024
00196 // TODO: add some more documentation
00197 namespace L4 { namespace Ipc {
00198
00215 struct L4_EXPORT Call
00216 {
00217 enum { Is_call = true };
00218 enum { Rights = 0 };
00219 static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00220 };
00221
00225 struct L4_EXPORT Call_zero_send_timeout : Call
00226 {
00227 static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; }
00228 };
00229
00245 template<unsigned RIGHTS>
00246 struct L4_EXPORT Call_t : Call
00247 {
00248 enum { Rights = RIGHTS };
00249 };
00250
00263 struct L4_EXPORT Send_only
00264 {
00265 enum { Is_call = false };
00266 enum { Rights = 0 };
00267 static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00268 };
00269
00270 namespace Msg {
00271
00282 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00283 struct L4_EXPORT Rpc_inline_call;
00284
00289 template<typename OP, typename CLASS, typename FLAGS, typename R,
00290 typename ...ARGS>
00291 struct L4_EXPORT Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00292 {
00293 template<typename T> struct Result { typedef T result_type; };
00294 enum
00295 {
00296 Return_tag = L4::Types::Same<R, l4_msgtag_t>::value
00297 };
00298
00300 typedef Rpc_inline_call type;
00302 typedef OP op_type;
00304 typedef CLASS class_type;
00306 typedef typename Result<R>::result_type result_type;
00308 typedef R ipc_type (ARGS...);
00310 typedef result_type func_type (typename _Elem<ARGS>::arg_type...);
00311
00313 typedef FLAGS flags_type;
00314
00315 template<typename RES>
00316 static typename L4::Types::Enable_if< Return_tag, RES >::type

```

```

00317 return_err(long err) { return l4_msgtag(err, 0, 0, 0); }
00318
00319 template<typename RES>
00320 static typename L4::Types::Enable_if< Return_tag, RES >::type
00321 return_ipc_err(l4_msgtag_t tag, l4_utcb_t const *) { return tag; }
00322
00323 template<typename RES>
00324 static typename L4::Types::Enable_if< Return_tag, RES >::type
00325 return_code(l4_msgtag_t tag) { return tag; }
00326
00327 template<typename RES>
00328 static typename L4::Types::Enable_if< !Return_tag, RES >::type
00329 return_err(long err) { return err; }
00330
00331 template<typename RES>
00332 static typename L4::Types::Enable_if< !Return_tag, RES >::type
00333 return_ipc_err(l4_msgtag_t, l4_utcb_t *utcb)
00334 { return l4_ipc_to_errno(l4_ipc_error_code(utcb)); }
00335
00336 template<typename RES>
00337 static typename L4::Types::Enable_if< !Return_tag, RES >::type
00338 return_code(l4_msgtag_t tag) { return tag.label(); }
00339
00340 static R call(L4::Cap<class_type> cap,
00341 typename _Elem<ARGS>::arg_type ...a,
00342 l4_utcb_t *utcb = l4_utcb()) throw();
00343 };
00344
00349 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00350 struct L4_EXPORT Rpc_call;
00351
00359 template<typename IPC, typename SIG> struct _Call;
00360
00362 template<typename IPC, typename R, typename ...ARGS>
00363 struct _Call<IPC, R (ARGS...)>
00364 {
00365 public:
00366 typedef typename IPC::class_type class_type;
00367 typedef typename IPC::result_type result_type;
00368
00369 private:
00370 L4::Cap<class_type> cap() const
00371 {
00372 return L4::Cap<class_type>(reinterpret_cast<l4_cap_idx_t>(this)
00373 & L4_CAP_MASK);
00374 }
00375
00376 public:
00377 result_type operator () (ARGS ...a, l4_utcb_t *utcb = l4_utcb()) const throw()
00378 { return IPC::call(cap(), a..., utcb); }
00379 };
00380
00381 template<typename IPC> struct Call : _Call<IPC, typename IPC::func_type> {};
00382
00394 template<typename OP,
00395 typename CLASS,
00396 typename FLAGS,
00397 typename R,
00398 typename ...ARGS>
00399 struct L4_EXPORT Rpc_call<OP, CLASS, R (ARGS...), FLAGS> :
00400 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00401 {
00402 static R call(L4::Cap<CLASS> cap,
00403 typename _Elem<ARGS>::arg_type ...a,
00404 l4_utcb_t *utcb = l4_utcb()) throw();
00405 };
00406
00407 #define L4_INLINE_RPC_SRV_FORWARD(name)
00408 template<typename OBJ> struct fwd
00409 {
00410 OBJ *o;
00411 fwd(OBJ *o) : o(o) {}
00412 template<typename ...ARGS> long call(ARGS ...a)
00413 { return o->op_##name(a...); }
00414 }
00415
00416 #define L4_INLINE_RPC_NF(res, name, args...)
00417 struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00418 {
00419 typedef L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args> type;
00420 L4_INLINE_RPC_SRV_FORWARD(name);
00421 }
00422
00423 #define L4_INLINE_RPC_NF_OP(op, res, name, args...)
00424 struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00425 {

```

```

00445 typedef L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args> type; \
00446 enum { Opcode = (op) }; \
00447 L4_INLINE_RPC_SRV_FORWARD(name); \
00448 }
00449
00450 #ifdef DOXYGEN
00451
00452 #define L4_INLINE_RPC(res, name, args, attr...) res name args
00453 #else
00454 #define L4_INLINE_RPC(res, name, args...) \
00455 L4_INLINE_RPC_NF(res, name, args); L4::IpC::Msg::Call<name##_t> name
00456 #endif
00457
00458 #ifdef DOXYGEN
00459
00460 #define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args
00461 #else
00462 #define L4_INLINE_RPC_OP(op, res, name, args...) \
00463 L4_INLINE_RPC_NF_OP(op, res, name, args); L4::IpC::Msg::Call<name##_t> name
00464 #endif
00465
00466 #define L4_RPC_NF(res, name, args...) \
00467 struct name##_t : L4::IpC::Msg::Rpc_call<name##_t, Class, res args> \
00468 { \
00469 typedef L4::IpC::Msg::Rpc_call<name##_t, Class, res args> type; \
00470 L4_INLINE_RPC_SRV_FORWARD(name); \
00471 }
00472
00473 #define L4_RPC_NF_OP(op, res, name, args...) \
00474 struct name##_t : L4::IpC::Msg::Rpc_call<name##_t, Class, res args> \
00475 { \
00476 typedef L4::IpC::Msg::Rpc_call<name##_t, Class, res args> type; \
00477 enum { Opcode = (op) }; \
00478 L4_INLINE_RPC_SRV_FORWARD(name); \
00479 }
00480
00481 #ifdef DOXYGEN
00482
00483 #define L4_RPC(res, name, args, attr...) res name args
00484 #else
00485 #define L4_RPC(res, name, args...) \
00486 L4_RPC_NF(res, name, args); L4::IpC::Msg::Call<name##_t> name
00487 #endif
00488
00489 #ifdef DOXYGEN
00490
00491 #define L4_RPC_OP(op, res, name, args, attr...) res name args
00492 #else
00493 #define L4_RPC_OP(op, res, name, args...) \
00494 L4_RPC_NF_OP(op, res, name, args); L4::IpC::Msg::Call<name##_t> name
00495 #endif
00496
00497 namespace Detail {
00498
00499 template<typename ...ARGS>
00500 struct Buf
00501 {
00502 public:
00503 template<typename DIR>
00504 static int write(char *, int offset, int)
00505 { return offset; }
00506
00507 template<typename DIR>
00508 static int read(char *, int offset, int, long)
00509 { return offset; }
00510
00511 typedef void Base;
00512 };
00513
00514 template<typename A, typename ...M>
00515 struct Buf<A, M...> : Buf<M...>
00516 {
00517 typedef Buf<M...> Base;
00518
00519 typedef Clnt_xmit<A> xmit;
00520 typedef typename _Elem<A>::arg_type arg_type;
00521 typedef Detail::_Plain<arg_type> plain;
00522
00523 template<typename DIR>
00524 static int
00525 write(char *base, int offset, int limit,
00526 arg_type a, typename _Elem<M>::arg_type ...m)
00527 {
00528 offset = xmit::to_msg(base, offset, limit, plain::deref(a),
00529 typename DIR::dir(), typename DIR::cls());
00530 return Base::template write<DIR>(base, offset, limit, m...);
00531 }
00532 }

```

```

00580 }
00581
00582 template<typename DIR>
00583 static int
00584 read(char *base, int offset, int limit, long ret,
00585 arg_type a, typename _Elem<M>::arg_type ...)
00586 {
00587 int r = xmit::from_msg(base, offset, limit, ret, plain::deref(a),
00588 typename DIR::dir(), typename DIR::cls());
00589 if (L4_LIKELY(r >= 0))
00590 return Base::template read<DIR>(base, r, limit, ret, m...);
00591
00592 if (_Elem<A>::Is_optional)
00593 return Base::template read<DIR>(base, offset, limit, ret, m...);
00594
00595 return r;
00596 }
00597 };
00598
00599 template <typename ...ARGS> struct _Part
00600 {
00601 typedef Buf<ARGS...> Data;
00602
00603 template<typename DIR>
00604 static int write(void *b, int offset, int limit,
00605 typename _Elem<ARGS>::arg_type ...)
00606 {
00607 int r = Data::template write<DIR>((char *)b, offset, limit, m...);
00608 if (L4_LIKELY(r >= offset))
00609 return r - offset;
00610 return r;
00611 }
00612
00613 template<typename DIR>
00614 static int read(void *b, int offset, int limit, long ret,
00615 typename _Elem<ARGS>::arg_type ...)
00616 {
00617 int r = Data::template read<DIR>((char *)b, offset, limit, ret, m...);
00618 if (L4_LIKELY(r >= offset))
00619 return r - offset;
00620 return r;
00621 }
00622 }
00623 };
00624
00631 template<typename IPC_TYPE, typename OPCODE = void>
00632 struct Part;
00633
00634 // The version without an op-code
00635 template<typename R, typename ...ARGS>
00636 struct Part<R (ARGS...), void> : _Part<ARGS...>
00637 {
00638 typedef Buf<ARGS...> Data;
00639
00640 // write arguments, skipping the dummy opcode
00641 template<typename DIR>
00642 static int write_op(void *b, int offset, int limit,
00643 int /*placeholder for op*/,
00644 typename _Elem<ARGS>::arg_type ...)
00645 {
00646 int r = Data::template write<DIR>((char *)b, offset, limit, m...);
00647 if (L4_LIKELY(r >= offset))
00648 return r - offset;
00649 return r;
00650 }
00651 }
00652 };
00653
00654 // Message part with additional opcode
00655 template<typename OPCODE, typename R, typename ...ARGS>
00656 struct Part<R (ARGS...), OPCODE> : _Part<ARGS...>
00657 {
00658 typedef OPCODE opcode_type;
00659 typedef Buf<opcode_type, ARGS...> Data;
00660
00661 // write arguments, including the opcode
00662 template<typename DIR>
00663 static int write_op(void *b, int offset, int limit,
00664 opcode_type op, typename _Elem<ARGS>::arg_type ...)
00665 {
00666 int r = Data::template write<DIR>((char *)b, offset, limit, op, m...);
00667 if (L4_LIKELY(r >= offset))
00668 return r - offset;
00669 return r;
00670 }
00671 }
00672 };
00673
00674
00675 } // namespace Detail

```

```

00676
00677 //-----
00678 // Implementation of the RPC call
00679 // TODO: Add support for timeout via special RPC argument
00680 // TODO: Add support for passing the UTCB pointer as argument
00681 //
00682 template<typename OP, typename CLASS, typename FLAGS, typename R,
00683 typename ...ARGS>
00684 inline R
00685 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>::
00686 call(L4::Cap<CLASS> cap,
00687 typename _Elem<ARGS>::arg_type ...a,
00688 l4_utcb_t *utcb) throw()
00689 {
00690 using namespace Ipc::Msg;
00691
00692 typedef typename Kobject_typeid<CLASS>::Iface::Rpcs Rpcs;
00693 typedef typename Rpcs::template Rpc<OP> Opt;
00694 typedef Detail::Part<ipc_type, typename Rpcs::opcode_type> Args;
00695
00696 l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00697
00698 // handle in-data part of the arguments
00699 int send_bytes =
00700 Args::template write_op<Do_in_data>(mrs->mr, 0, Mr_bytes,
00701 Opt::Opcode, a...);
00702
00703 if (L4_UNLIKELY(send_bytes < 0))
00704 return return_err<R>(send_bytes);
00705
00706 send_bytes = align_to<l4_umword_t>(send_bytes);
00707 int const send_words = send_bytes / Word_bytes;
00708 // write the in-items part of the message if there is one
00709 int item_bytes =
00710 Args::template write<Do_in_items>(&mrs->mr[send_words], 0,
00711 Mr_bytes - send_bytes, a...);
00712
00713 if (L4_UNLIKELY(item_bytes < 0))
00714 return return_err<R>(item_bytes);
00715
00716 int send_items = item_bytes / Item_bytes;
00717
00718 {
00719 // setup the receive buffers for the RPC call
00720 l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
00721 // XXX: we currently support only one type of receive buffers per call
00722 brs->bdr = 0; // we always start at br[0]
00723
00724 // the limit leaves us at least one register for the zero terminator
00725 // add the buffers given as arguments to the buffer registers
00726 int bytes =
00727 Args::template write<Do_rcv_buffers>(brs->br, 0, Br_bytes - Word_bytes,
00728 a...);
00729
00730 if (L4_UNLIKELY(bytes < 0))
00731 return return_err<R>(bytes);
00732
00733 brs->br[bytes / Word_bytes] = 0;
00734 }
00735
00736
00737 // here we do the actual IPC -----
00738 l4_msgtag_t t;
00739 t = l4_msgtag(CLASS::Protocol, send_words, send_items, 0);
00740 // do the call (Q: do we need support for timeouts?)
00741 if (flags_type::Is_call)
00742 t = l4_ipc_call(cap.cap(), utcb, t, flags_type::timeout());
00743 else
00744 {
00745 t = l4_ipc_send(cap.cap(), utcb, t, flags_type::timeout());
00746 if (L4_UNLIKELY(t.has_error()))
00747 return return_ipc_err<R>(t, utcb);
00748
00749 return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
00750 }
00751
00752 // unmarshalling starts here -----
00753
00754 // bail out early in the case of an IPC error
00755 if (L4_UNLIKELY(t.has_error()))
00756 return return_ipc_err<R>(t, utcb);
00757
00758 // take the label as return value
00759 long r = t.label();
00760
00761 // bail out on negative error codes too
00762 if (L4_UNLIKELY(r < 0))

```



## Data Structures

- class [L4::Server\\_object](#)  
*Abstract server object to be used with [L4::Server](#) and [L4::Basic\\_registry](#).*
- struct [L4::Server\\_object\\_t](#) < IFACE, BASE >  
*Base class (template) for server implementing server objects.*
- struct [L4::Server\\_object\\_x](#) < Derived, IFACE, BASE >  
*Helper class to implement p\_dispatch based server objects.*
- struct [L4::Irq\\_handler\\_object](#)  
*Server object base class for handling IRQ messages.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 15.287.1 Detailed Description

IPC server loop.

Definition in file [ipc\\_server](#).

## 15.288 ipc\_server

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/sys/typeinfo_svr>
00029 #include <l4/sys/err.h>
00030 #include <l4/cxx/ipc_stream>
00031 #include <l4/sys/cxx/ipc_epiface>
00032 #include <l4/sys/cxx/ipc_server_loop>
00033 #include <l4/cxx/type_traits>
00034 #include <l4/cxx/exceptions>
00035
00036 namespace L4 {
00037
00049 class Server_object : public Epiface
00050 {
00051 public:
00069 virtual int dispatch(unsigned long rights, Ipc::Iostream &ios) = 0;
00070
00071 l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights,
00072 l4_utcb_t *utcb)
00073 {

```

```

00073 L4::Ipc::Iostream ios(utcb);
00074 ios.tag() = tag;
00075 int r = dispatch(rights, ios);
00076 return ios.prepare_ipc(r);
00077 }
00078
00079 Cap<Kobject> obj_cap() const
00080 { return cap_cast<Kobject>(Epiface::obj_cap()); }
00081 };
00082
00090 template<typename IFACE, typename BASE = L4::Server_object>
00091 struct Server_object_t : BASE
00092 {
00093 typedef IFACE Interface;
00094
00095 typename BASE::Demand get_buffer_demand() const
00096 { return typename L4::Kobject_typeid<IFACE>::Demand(); }
00097
00098 int dispatch_meta_request(L4::Ipc::Iostream &ios)
00099 { return L4::Util::handle_meta_request<IFACE>(ios); }
00100
00101 template<typename THIS>
00102 static int proto_dispatch(THIS *self, l4_umword_t rights,
00103 L4::Ipc::Iostream &ios)
00104 {
00105 l4_msgtag_t t;
00106 ios >> t;
00107 return Kobject_typeid<IFACE>::proto_dispatch(self, t.label(),
00108 rights, ios);
00109 }
00110 };
00111
00112 template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
00113 struct Server_object_x : Server_object_t<IFACE, BASE>
00114 {
00115 int dispatch(l4_umword_t r, L4::Ipc::Iostream &ios)
00116 {
00117 return Server_object_t<IFACE, BASE>::proto_dispatch(
00118 static_cast<Derived *>(this),
00119 r, ios);
00120 }
00121 };
00122
00123 struct Irq_handler_object : Server_object_t<Kobject> {};
00124
00125

```

## 15.289 l4/sys/cxx/ipc\_types File Reference

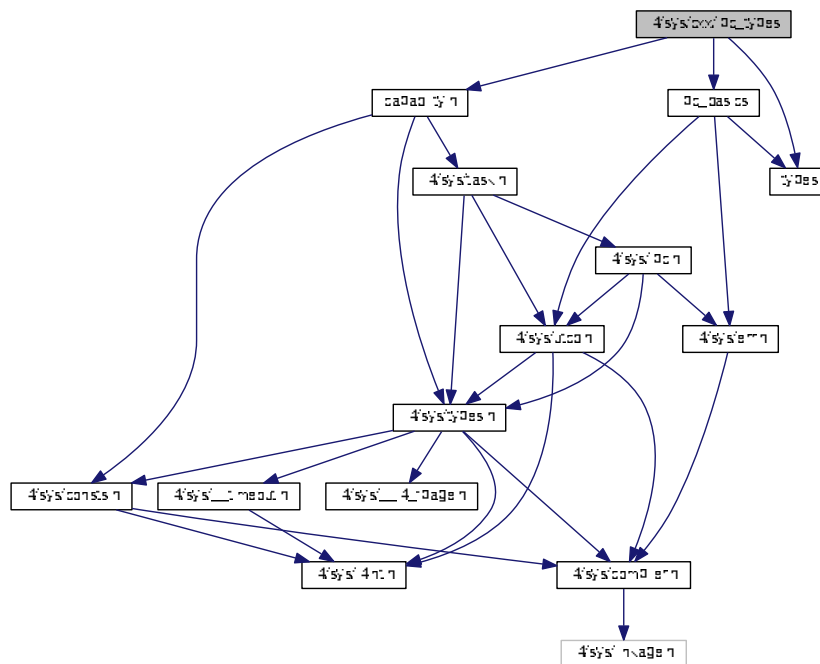
```

#include "capability.h"
#include "types"
#include "ipc_basics"

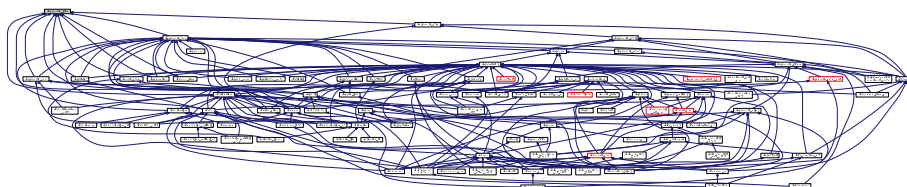
```



Include dependency graph for ipc\_types:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `L4::lpc::Out< T >`  
*Mark an argument as a output value in an RPC signature.*
- struct `L4::lpc::In_out< T >`  
*Mark an argument as in-out argument.*
- struct `L4::lpc::As_value< T >`  
*Pass the argument as plain data value.*
- struct `L4::lpc::Opt< T >`  
*Attribute for defining an optional RPC argument.*
- class `L4::lpc::Small_buf`  
*A receive item for receiving a single capability.*
- class `L4::lpc::Snd_item`  
*RPC wrapper for a send item.*
- class `L4::lpc::Buf_item`

- RPC warpper for a receive item.*
- class [L4::lpc::Gen\\_fpage< T >](#)  
*Generic RPC wrapper for [L4](#) flex-pages.*
- class [L4::lpc::Cap< T >](#)  
*Capability type for RPC interfaces (see [L4::Cap< T >](#)).*

## Namespaces

- [L4](#)  
*[L4](#) low-level kernel interface.*
- [L4::lpc](#)  
*IPC related functionality.*
- [L4::lpc::Msg](#)  
*IPC Message related functionality.*

## Typedefs

- typedef Gen\_fpage< Snd\_item > [L4::lpc::Snd\\_fpage](#)  
*Send flex-page.*
- typedef Gen\_fpage< Buf\_item > [L4::lpc::Rcv\\_fpage](#)  
*Rcv flex-page.*

## Functions

- template<typename T >  
 Cap< T > [L4::lpc::make\\_cap](#) ([L4::Cap< T >](#) cap, unsigned rights)  
*Make an [L4::lpc::Cap< T >](#) for the given capability and rights.*
- template<typename T >  
 Cap< T > [L4::lpc::make\\_cap\\_rw](#) ([L4::Cap< T >](#) cap)  
*Make an [L4::lpc::Cap< T >](#) for the given capability with [L4\\_CAP\\_FPAGE\\_RW](#) rights.*
- template<typename T >  
 Cap< T > [L4::lpc::make\\_cap\\_rws](#) ([L4::Cap< T >](#) cap)  
*Make an [L4::lpc::Cap< T >](#) for the given capability with [L4\\_CAP\\_FPAGE\\_RWS](#) rights.*
- template<typename T >  
 Cap< T > [L4::lpc::make\\_cap\\_full](#) ([L4::Cap< T >](#) cap)  
*Make an [L4::lpc::Cap< T >](#) for the given capability with full fpage and object-specific rights.*

## 15.290 ipc\_types

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however

```

```

00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019
00020 #include "capability.h"
00021 #include "types"
00022 #include "ipc_basics"
00027 namespace L4 {
00028
00030 typedef int Opcode;
00031
00032 namespace Ipc {
00033
00042 template<typename T> struct L4_EXPORT Out;
00043
00044
00052 template<typename T> struct L4_EXPORT In_out
00053 {
00054 T v;
00055 In_out() {}
00056 In_out(T v) : v(v) {}
00057 operator T () const { return v; }
00058 operator T & () { return v; }
00059 };
00060
00061 namespace Msg {
00062 template<typename A> struct Elem< In_out<A *> > : Elem<A *> {};
00063
00064 template<typename A>
00065 struct Svr_xmit< In_out<A *> > : Svr_xmit<A *>, Svr_xmit<A const *>
00066 {
00067 using Svr_xmit<A *>::from_svr;
00068 using Svr_xmit<A const *>::to_svr;
00069 };
00070
00071 template<typename A>
00072 struct Clnt_xmit< In_out<A *> > : Clnt_xmit<A *>, Clnt_xmit<A const *>
00073 {
00074 using Clnt_xmit<A *>::from_msg;
00075 using Clnt_xmit<A const *>::to_msg;
00076 };
00077
00078 template<typename A>
00079 struct Is_valid_rpc_type< In_out<A *> > : Is_valid_rpc_type<A *> {};
00080 template<typename A>
00081 struct Is_valid_rpc_type< In_out<A const *> > : L4::Types::False {};
00082
00083 #ifdef CONFIG_ALLOW_REFS
00084 template<typename A> struct Elem< In_out<A &> > : Elem<A &> {};
00085
00086 template<typename A>
00087 struct Svr_xmit< In_out<A &> > : Svr_xmit<A &>, Svr_xmit<A const &>
00088 {
00089 using Svr_xmit<A &>::from_svr;
00090 using Svr_xmit<A const &>::to_svr;
00091 };
00092
00093 template<typename A>
00094 struct Clnt_xmit< In_out<A &> > : Clnt_xmit<A &>, Clnt_xmit<A const &>
00095 {
00096 using Clnt_xmit<A &>::from_msg;
00097 using Clnt_xmit<A const &>::to_msg;
00098 };
00099
00100 template<typename A>
00101 struct Is_valid_rpc_type< In_out<A &> > : Is_valid_rpc_type<A &> {};
00102 template<typename A>
00103 struct Is_valid_rpc_type< In_out<A const &> > : L4::Types::False {};
00104
00105 #else
00106
00107 template<typename A>
00108 struct Is_valid_rpc_type< In_out<A &> > : L4::Types::False {};
00109
00110 #endif
00111
00112 // Value types don't make sense for output.
00113 template<typename A>
00114 struct Is_valid_rpc_type< In_out<A> > : L4::Types::False {};
00115
00116 }
00117
00118
00127 template<typename T> struct L4_EXPORT As_value
00128 {
00129 typedef T value_type;

```

```

00130 T v;
00131 As_value() {}
00132 As_value(T v) : v(v) {}
00133 operator T () const { return v; }
00134 operator T & () { return v; }
00135 };
00136
00137 namespace Msg {
00138 template<typename T> struct Class< As_value<T> > : Cls_data {};
00139 template<typename T> struct Elem< As_value<T> > : Elem<T> {};
00140 template<typename T> struct Elem< As_value<T> *> : Elem<T *> {};
00141 }
00142
00143
00147 template<typename T> struct L4_EXPORT Opt
00148 {
00149 T _value;
00150 bool _valid;
00151
00153 Opt() : _valid(false) {}
00154
00156 Opt(T value) : _value(value), _valid(true) {}
00157
00159 Opt &operator = (T value)
00160 {
00161 this->_value = value;
00162 this->_valid = true;
00163 return *this;
00164 }
00165
00167 void set_valid(bool valid = true) { _valid = valid; }
00168
00170 T *operator -> () { return &this->_value; }
00172 T const *operator -> () const { return &this->_value; }
00174 T value() const { return this->_value; }
00176 T &value() { return this->_value; }
00178 bool is_valid() const { return this->_valid; }
00179 };
00180
00181 namespace Msg {
00182 template<typename T> struct Elem< Opt<T &> > : Elem<T &>
00183 {
00184 enum { Is_optional = true };
00185 typedef Opt<typename Elem<T &>::svr_type> &svr_arg_type;
00186 typedef Opt<typename Elem<T &>::svr_type> svr_type;
00187 };
00188
00189 template<typename T> struct Elem< Opt<T *> > : Elem<T *>
00190 {
00191 enum { Is_optional = true };
00192 typedef Opt<typename Elem<T *>::svr_type> &svr_arg_type;
00193 typedef Opt<typename Elem<T *>::svr_type> svr_type;
00194 };
00195
00196
00197
00198 template<typename T, typename CLASS>
00199 struct Svr_val_ops<Opt<T>, Dir_out, CLASS> : Svr_noops< Opt<T> >
00200 {
00201 typedef Opt<T> svr_type;
00202 typedef Svr_val_ops<T, Dir_out, CLASS> Native;
00203
00204 using Svr_noops< Opt<T> >::to_svr;
00205 static int to_svr(char *msg, unsigned offset, unsigned limit,
00206 Opt<T> &arg, Dir_out, CLASS)
00207 {
00208 return Native::to_svr(msg, offset, limit, arg.value(), Dir_out(), CLASS());
00209 }
00210
00211 using Svr_noops< Opt<T> >::from_svr;
00212 static int from_svr(char *msg, unsigned offset, unsigned limit, long ret,
00213 svr_type &arg, Dir_out, CLASS)
00214 {
00215 if (arg.is_valid())
00216 return Native::from_svr(msg, offset, limit, ret, arg.value(),
00217 Dir_out(), CLASS());
00218 return offset;
00219 }
00220 };
00221
00222 template<typename T> struct Elem< Opt<T> > : Elem<T>
00223 {
00224 enum { Is_optional = true };
00225 typedef Opt<T> arg_type;
00226 };
00227
00228 template<typename T> struct Elem< Opt<T const *> > : Elem<T const *>

```

```

00229 {
00230 enum { Is_optional = true };
00231 typedef Opt<T const *> arg_type;
00232 };
00233
00234 template<typename T>
00235 struct Is_valid_rpc_type< Opt<T const &> > : L4::Types::False {};
00236
00237 template<typename T, typename CLASS>
00238 struct Clnt_val_ops<Opt<T>, Dir_in, CLASS> : Clnt_noops< Opt<T> >
00239 {
00240 typedef Opt<T> arg_type;
00241 typedef Detail::_Clnt_val_ops<typename Elem<T>::arg_type, Dir_in, CLASS> Native;
00242
00243 using Clnt_noops< Opt<T> >::to_msg;
00244 static int to_msg(char *msg, unsigned offset, unsigned limit,
00245 arg_type arg, Dir_in, CLASS)
00246 {
00247 if (arg.is_valid())
00248 return Native::to_msg(msg, offset, limit,
00249 Detail::_Plain<T>::deref(arg.value()),
00250 Dir_in(), CLASS());
00251 return offset;
00252 }
00253 };
00254
00255 template<typename T> struct Class< Opt<T> > :
00256 Class< typename Detail::_Plain<T>::type > {};
00257 template<typename T> struct Direction< Opt<T> > : Direction<T> {};
00258 }
00259
00260 class L4_EXPORT Small_buf
00261 {
00262 public:
00263 explicit Small_buf(L4::Cap<void> cap, unsigned long flags = 0)
00264 : _data(cap.cap() | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00265
00266 explicit Small_buf(l4_cap_idx_t cap, unsigned long flags = 0)
00267 : _data(cap | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00268
00269 l4_umword_t raw() const { return _data; }
00270 private:
00271 l4_umword_t _data;
00272 };
00273
00274 class Snd_item
00275 {
00276 public:
00277 Snd_item(l4_umword_t base, l4_umword_t data) : _base(base), _data(data) {}
00278
00279 protected:
00280 l4_umword_t _base;
00281 l4_umword_t _data;
00282 };
00283
00284 class Buf_item
00285 {
00286 public:
00287 Buf_item(l4_umword_t base, l4_umword_t data) : _base(base), _data(data) {}
00288
00289 protected:
00290 l4_umword_t _base;
00291 l4_umword_t _data;
00292 };
00293
00294 template< typename T >
00295 class L4_EXPORT Gen_fpage : public T
00296 {
00297 public:
00298 enum Type
00299 {
00300 Special = L4_FPAGE_SPECIAL << 4,
00301 Memory = L4_FPAGE_MEMORY << 4,
00302 Io = L4_FPAGE_IO << 4,
00303 Obj = L4_FPAGE_OBJ << 4
00304 };
00305
00306 enum Map_type
00307 {
00308 Map = L4_MAP_ITEM_MAP,
00309 Grant = L4_MAP_ITEM_GRANT,
00310 };
00311
00312 enum Cacheopt
00313 {
00314 None = 0,
00315 Cached = L4_FPAGE_CACHEABLE << 4,

```

```

00345 Buffered = L4_FPAGE_BUFFERABLE << 4,
00346 Uncached = L4_FPAGE_UNCACHEABLE << 4
00347 };
00348
00349 enum Continue
00350 {
00351 Single = 0,
00352 Last = 0,
00353 More = L4_ITEM_CONT,
00354 Compound = L4_ITEM_CONT,
00355 };
00356
00357 private:
00358 Gen_fpage(l4_umword_t d, l4_umword_t fp) : T(d, fp) {}
00359
00360 Gen_fpage(Type type, l4_addr_t base, int order,
00361 unsigned char rights,
00362 l4_addr_t snd_base,
00363 Map_type map_type,
00364 Cacheopt cache, Continue cont)
00365 : T(L4_ITEM_MAP | (snd_base & (~0UL << 10)) | l4_umword_t(map_type) |
00366 l4_umword_t(cache)
00367 | l4_umword_t(cont),
00368 base | l4_umword_t(type) | rights | (l4_umword_t(order) << 6))
00369 {}
00370 public:
00371 Gen_fpage() : T(0, 0) {}
00372 Gen_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00373 Map_type map_type = Map,
00374 Cacheopt cache = None, Continue cont = Last)
00375 : T(L4_ITEM_MAP | (snd_base & (~0UL << 10)) | l4_umword_t(map_type) |
00376 l4_umword_t(cache)
00377 | l4_umword_t(cont),
00378 fp.raw)
00379 {}
00380 Gen_fpage(L4::Cap<void> cap, unsigned rights, Map_type map_type = Map)
00381 : T(L4_ITEM_MAP | l4_umword_t(map_type) | (rights & 0xf0),
00382 cap.fpage(rights).raw)
00383 {}
00384
00385 static Gen_fpage<T> obj(l4_addr_t base, int order,
00386 unsigned char rights,
00387 l4_addr_t snd_base = 0,
00388 Map_type map_type = Map,
00389 Continue cont = Last)
00390 {
00391 return Gen_fpage<T>(Obj, base << 12, order, rights, snd_base, map_type, None, cont);
00392 }
00393
00394 static Gen_fpage<T> mem(l4_addr_t base, int order,
00395 unsigned char rights,
00396 l4_addr_t snd_base = 0,
00397 Map_type map_type = Map,
00398 Cacheopt cache = None, Continue cont = Last)
00399 {
00400 return Gen_fpage<T>(Memory, base, order, rights, snd_base,
00401 map_type, cache, cont);
00402 }
00403
00404 static Gen_fpage<T> rmem(l4_addr_t base, int order,
00405 l4_addr_t snd_base,
00406 unsigned char rights, unsigned cap_br)
00407 {
00408 return Gen_fpage<T>(
00409 L4_ITEM_MAP | (snd_base & (~0UL << 10)) | l4_umword_t(Map)
00410 | l4_umword_t(None) | l4_umword_t(Compound) | (cap_br << 8),
00411 base | l4_umword_t(Memory) | rights | (l4_umword_t(order) << 6));
00412 }
00413
00414 static Gen_fpage<T> io(l4_addr_t base, int order,
00415 unsigned char rights,
00416 l4_addr_t snd_base = 0,
00417 Map_type map_type = Map,
00418 Continue cont = Last)
00419 {
00420 return Gen_fpage<T>(Io, base << 12, order, rights, snd_base, map_type, None, cont);
00421 }
00422
00423 unsigned order() const { return (T::_data >> 6) & 0x3f; }
00424 unsigned snd_order() const { return (T::_data >> 6) & 0x3f; }
00425 unsigned rcv_order() const { return (T::_base >> 6) & 0x3f; }
00426 l4_addr_t base() const { return T::_data & (~0UL << 12); }
00427 l4_addr_t snd_base() const { return T::_base & (~0UL << 10); }
00428 void snd_base(l4_addr_t b) { T::_base = (T::_base & ~(~0UL << 10)) | (b & (~0UL << 10)); }

```

```

00429
00431 bool is_valid() const { return T::_base & L4_ITEM_MAP; }
00438 bool cap_received() const { return (T::_base & 0x3e) == 0x38; }
00450 bool id_received() const { return (T::_base & 0x3e) == 0x3c; }
00460 bool local_id_received() const { return (T::_base & 0x3e) == 0x3e; }
00461
00468 bool is_compound() const { return T::_base & 1; }
00470 l4_umword_t data() const { return T::_data; }
00472 l4_umword_t base_x() const { return T::_base; }
00473 };
00474
00475
00477 typedef Gen_fpage<Snd_item> Snd_fpage;
00479 typedef Gen_fpage<Buf_item> Rcv_fpage;
00480
00481 #ifdef L4_CXX_IPC_SUPPORT_STRINGS
00482 template <typename T, typename B>
00483 class Gen_string : public T
00484 {
00485 public:
00486 Gen_string() : T(0, 0) {}
00487 Gen_string(B buf, unsigned long size)
00488 : T(size << 10, l4_umword_t(buf))
00489 {}
00490
00491 unsigned long len() const { return T::_base >> 10; }
00492 };
00493
00494 typedef Gen_string<Snd_item, void const *> Snd_string;
00495 typedef Gen_string<Buf_item, void *> Rcv_string;
00496 #endif
00497
00498
00499 namespace Msg {
00500
00501 // Snd_fpage are out items
00502 template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00503
00504 // Rcv_fpage are buffer items
00505 template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00506
00507 // Remove receive buffers from server-side arguments
00508 template<> struct Elem<L4::Ipc::Rcv_fpage>
00509 {
00510 typedef L4::Ipc::Rcv_fpage arg_type;
00511 typedef void svr_type;
00512 typedef void svr_arg_type;
00513 enum { Is_optional = false };
00514 };
00515
00516 // Rcv_fpage are buffer items
00517 template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00518
00519 // Remove receive buffers from server-side arguments
00520 template<> struct Elem<L4::Ipc::Small_buf>
00521 {
00522 typedef L4::Ipc::Small_buf arg_type;
00523 typedef void svr_type;
00524 typedef void svr_arg_type;
00525 enum { Is_optional = false };
00526 };
00527 } // namespace Msg
00528
00529 // L4::Cap<> handling
00530
00541 template<typename T> class Cap
00542 {
00543 template<typename O> friend class Cap;
00544 l4_umword_t _cap_n_rights;
00545
00546 public:
00547 enum
00548 {
00554 Rights_mask = 0xff,
00555
00560 Cap_mask = L4_CAP_MASK
00561 };
00562
00564 template<typename O>
00565 Cap(Cap<O> const &o) : _cap_n_rights(o._cap_n_rights)
00566 { T *x = (O*)1; (void)x; }
00567
00569 Cap(L4::Cap<T> cap)
00570 : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00571 {}
00572
00574 template<typename O>

```

```

00575 Cap(L4::Cap<O> cap)
00576 : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00577 { T *x = (O*)1; (void)x; }
00578
00580 Cap() : _cap_n_rights(L4_INVALID_CAP) {}
00581
00589 Cap(L4::Cap<T> cap, unsigned char rights)
00590 : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00591
00597 static Cap from_ci(l4_cap_idx_t c)
00598 { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00599
00601 L4::Cap<T> cap() const
00602 { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00603
00605 unsigned rights() const
00606 { return _cap_n_rights & Rights_mask; }
00607
00609 L4::Ipc::Snd_fpage fpage() const
00610 { return L4::Ipc::Snd_fpage(cap(), rights()); }
00611
00613 bool is_valid() const throw()
00614 { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00615 };
00616
00623 template<typename T>
00624 Cap<T> make_cap(L4::Cap<T> cap, unsigned rights)
00625 { return Cap<T>(cap, rights); }
00626
00633 template<typename T>
00634 Cap<T> make_cap_rw(L4::Cap<T> cap)
00635 { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00636
00643 template<typename T>
00644 Cap<T> make_cap_rws(L4::Cap<T> cap)
00645 { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00646
00661 template<typename T>
00662 Cap<T> make_cap_full(L4::Cap<T> cap)
00663 { return Cap<T>(cap, L4_CAP_FPAGE_RWS |
00664 L4_FPAGE_C_OBJ_RIGHTS); }
00664
00665 // caps are special the have an invalid representation
00666 template<typename T> struct L4_EXPORT Opt< Cap<T> >
00667 {
00668 Cap<T> _value;
00669 Opt() {}
00670 Opt(Cap<T> value) : _value(value) {}
00671 Opt(L4::Cap<T> value) : _value(value) {}
00672 Opt &operator = (Cap<T> value)
00673 { this->_value = value; }
00674 Opt &operator = (L4::Cap<T> value)
00675 { this->_value = value; }
00676
00677 Cap<T> value() const { return this->_value; }
00678 bool is_valid() const { return this->_value.is_valid(); }
00679 };
00680
00681 namespace Msg {
00682 // prohibit L4::Cap as argument
00683 template<typename A>
00684 struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00685
00686 template<typename A> struct Class< Cap<A> > : Cls_item {};
00687 template<typename A> struct Elem< Cap<A> >
00688 {
00689 enum { Is_optional = false };
00690 typedef Cap<A> arg_type;
00691 typedef L4::Ipc::Snd_fpage svr_type;
00692 typedef L4::Ipc::Snd_fpage svr_arg_type;
00693 };
00694
00695
00696 template<typename A, typename CLASS>
00697 struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
00698 Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
00699 {};
00700
00701 template<typename A, typename CLASS>
00702 struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
00703 Clnt_noops< Cap<A> >
00704 {
00705 using Clnt_noops< Cap<A> >::to_msg;
00706
00707 static int to_msg(char *msg, unsigned offset, unsigned limit,
00708 Cap<A> arg, Dir_in, Cls_item)

```



```

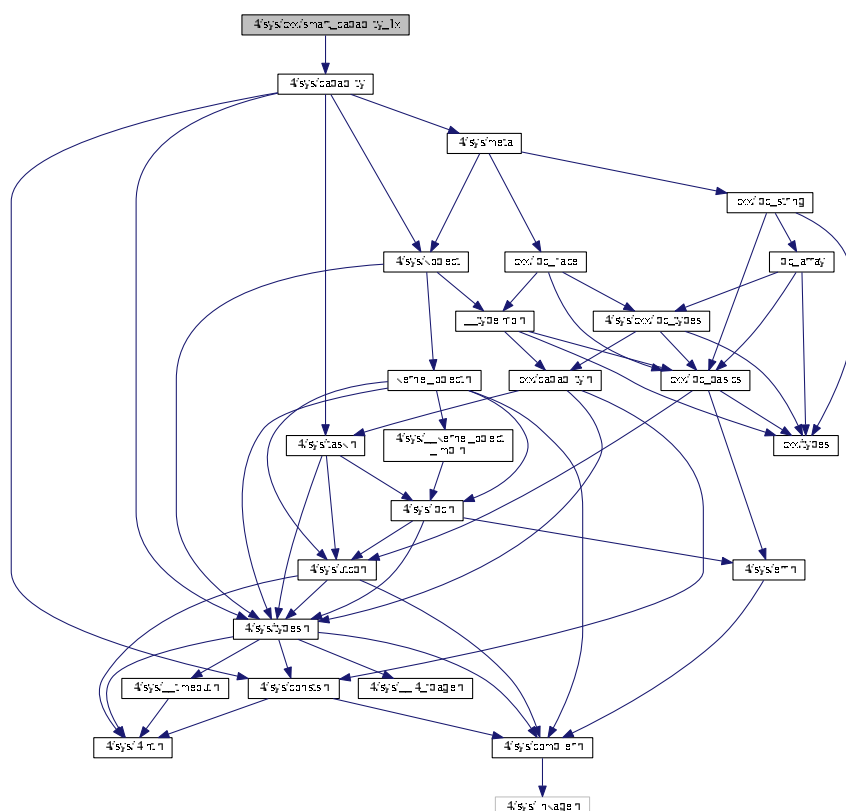
00710 {
00711 // passing an invalid cap as mandatory argument is an error
00712 // XXX: This checks for a client calling error, we could
00713 // also just ignore this for performance reasons and
00714 // let the client fail badly (Alex: I'd prefer this)
00715 if (L4_UNLIKELY(!arg.is_valid()))
00716 return -L4_MSGMISSARG;
00717
00718 return msg_add(msg, offset, limit, arg.fpage());
00719 }
00720 };
00721
00722 template<typename A>
00723 struct Elem<Out<L4::Cap<A> > >
00724 {
00725 enum { Is_optional = false };
00726 typedef L4::Cap<A> arg_type;
00727 typedef Ipc::Cap<A> svr_type;
00728 typedef svr_type &svr_arg_type;
00729 };
00730
00731 template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
00732 template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};
00733
00734 template<typename A>
00735 struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
00736 Clnt_noops< L4::Cap<A> >
00737 {
00738 using Clnt_noops< L4::Cap<A> >::to_msg;
00739 static int to_msg(char *msg, unsigned offset, unsigned limit,
00740 L4::Cap<A> arg, Dir_in, Cls_buffer)
00741 {
00742 if (L4_UNLIKELY(!arg.is_valid()))
00743 return -L4_MSGMISSARG; // no buffer inserted
00744 return msg_add(msg, offset, limit, Small_buf(arg));
00745 }
00746 };
00747
00748 template<typename A>
00749 struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
00750 Svr_noops<Cap<A> &>
00751 {
00752 using Svr_noops<Cap<A> &>::from_svr;
00753 static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00754 Cap<A> arg, Dir_out, Cls_item)
00755 {
00756 if (L4_UNLIKELY(!arg.is_valid()))
00757 // do not map anything
00758 return msg_add(msg, offset, limit, L4::Ipc::Snd_fpage(arg.
00759 cap(), 0));
00759
00760 return msg_add(msg, offset, limit, arg.fpage());
00761 }
00762 };
00763
00764 // prohibit a UTCB pointer as normal RPC argument
00765 template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};
00766
00767 } // namespace Msg
00768 } // namespace Ipc
00769 } // namespace L4
00770

```

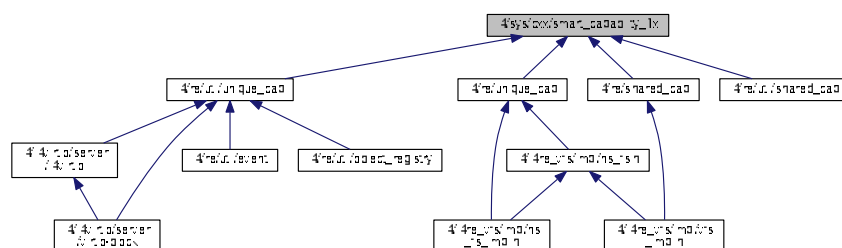
## 15.291 l4/sys/cxx/smart\_capability\_1x File Reference

```
#include <l4/sys/capability>
```

Include dependency graph for smart\_capability\_1x:



This graph shows which files directly or indirectly include this file:



## Namespaces

- **L4**  
*L4 low-level kernel interface.*

## 15.292 smart\_capability\_1x

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
```

```

00006 /*
00007 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026
00027 namespace L4 { namespace Detail {
00028
00029 template< typename T, typename IMPL >
00030 class Smart_cap_base : public Cap_base, protected IMPL
00031 {
00032 protected:
00033 template<typename X>
00034 static IMPL &impl(Smart_cap_base<X, IMPL> &o) { return o; }
00035
00036 template<typename X>
00037 static IMPL const &impl(Smart_cap_base<X, IMPL> const &o) { return o; }
00038
00039 public:
00040 template<typename X, typename I>
00041 friend class ::L4::Detail::Smart_cap_base;
00042
00043 Smart_cap_base(Smart_cap_base const &) = delete;
00044 Smart_cap_base &operator = (Smart_cap_base const &) = delete;
00045
00046 Smart_cap_base() noexcept : Cap_base(Invalid) {}
00047
00048 explicit Smart_cap_base(Cap_base::Cap_type t) noexcept
00049 : Cap_base(t)
00050 {}
00051
00052 template<typename O>
00053 explicit constexpr Smart_cap_base(Cap<O> c) noexcept
00054 : Cap_base(c.cap())
00055 {}
00056
00057 template<typename O>
00058 explicit constexpr Smart_cap_base(Cap<O> c, IMPL const &impl) noexcept
00059 : Cap_base(c.cap()), IMPL(impl)
00060 {}
00061
00062 Cap<T> release() noexcept
00063 {
00064 l4_cap_idx_t c = this->cap();
00065 IMPL::invalidate(*this);
00066 return Cap<T>(c);
00067 }
00068
00069 void reset()
00070 { IMPL::free(*this); }
00071
00072 Cap<T> operator -> () const noexcept { return Cap<T>(this->cap()); }
00073 Cap<T> get() const noexcept { return Cap<T>(this->cap()); }
00074 ~Smart_cap_base() noexcept { IMPL::free(*this); }
00075 };
00076
00077
00078 template< typename T, typename IMPL >
00079 class Unique_cap_impl final : public Smart_cap_base<T, IMPL>
00080 {
00081 private:
00082 typedef Smart_cap_base<T, IMPL> Base;
00083
00084 public:
00085 using Base::Base;
00086 Unique_cap_impl() noexcept = default;
00087
00088 Unique_cap_impl(Unique_cap_impl &&o) noexcept
00089 : Base(o.release(), Base::impl(o))
00090 {}
00091
00092 template<typename O>

```

```

00093 Unique_cap_impl(Unique_cap_impl<O, IMPL> &&o) noexcept
00094 : Base(o.release(), Base::impl(o))
00095 { T* __t = ((O*)100); (void)__t; }
00096
00097 Unique_cap_impl &operator = (Unique_cap_impl &&o) noexcept
00098 {
00099 if (&o == this)
00100 return *this;
00101
00102 IMPL::free(*this);
00103 this->_c = o.release().cap();
00104 this->IMPL::operator = (Base::impl(o));
00105 return *this;
00106 }
00107
00108 template<typename O>
00109 Unique_cap_impl &operator = (Unique_cap_impl<O, IMPL> &&o) noexcept
00110 {
00111 T* __t = ((O*)100); (void)__t;
00112
00113 IMPL::free(*this);
00114 this->_c = o.release().cap();
00115 this->IMPL::operator = (Base::impl(o));
00116 return *this;
00117 }
00118 };
00119
00120 template<typename T, typename IMPL>
00121 class Shared_cap_impl final : public Smart_cap_base<T, IMPL>
00122 {
00123 private:
00124 typedef Smart_cap_base<T, IMPL> Base;
00125
00126 public:
00127 using Base::Base;
00128 Shared_cap_impl() noexcept = default;
00129
00130 Shared_cap_impl(Shared_cap_impl &&o) noexcept
00131 : Base(o.release())
00132 {}
00133
00134 template<typename O>
00135 Shared_cap_impl(Shared_cap_impl<O, IMPL> &&o) noexcept
00136 : Base(o.release())
00137 { T* __t = ((O*)100); (void)__t; }
00138
00139 Shared_cap_impl &operator = (Shared_cap_impl &&o) noexcept
00140 {
00141 if (&o == this)
00142 return *this;
00143
00144 IMPL::free(*this);
00145 this->_c = o.release().cap();
00146 this->IMPL::operator = (Base::impl(o));
00147 return *this;
00148 }
00149
00150 template<typename O>
00151 Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> &&o) noexcept
00152 {
00153 T* __t = ((O*)100); (void)__t;
00154
00155 IMPL::free(*this);
00156 this->_c = o.release().cap();
00157 this->IMPL::operator = (Base::impl(o));
00158 return *this;
00159 }
00160
00161 Shared_cap_impl(Shared_cap_impl const &o) noexcept
00162 : Base(L4::Cap<T>(IMPL::copy(o).cap()))
00163 {}
00164
00165 template<typename O>
00166 Shared_cap_impl(Shared_cap_impl<O, IMPL> const &o) noexcept
00167 : Base(IMPL::copy(o))
00168 { T* __t = ((O*)100); (void)__t; }
00169
00170 Shared_cap_impl &operator = (Shared_cap_impl const &o) noexcept
00171 {
00172 if (&o == this)
00173 return *this;
00174
00175 IMPL::free(*this);
00176 this->IMPL::operator = (static_cast<IMPL const &>(o));
00177 this->_c = this->IMPL::copy(o).cap();
00178 return *this;
00179 }

```

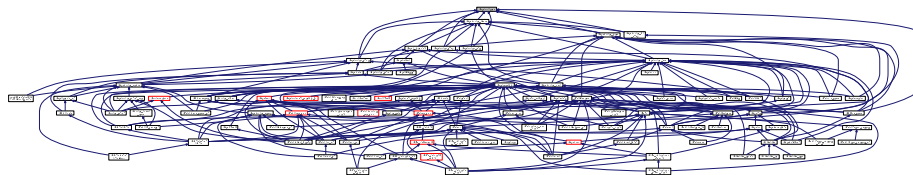
```

00180
00181 template<typename O>
00182 Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> const &o) noexcept
00183 {
00184 T* __t = ((O*)100); (void)__t;
00185 IMPL::free(*this);
00186 this->IMPL::operator = (static_cast<IMPL const &>(o));
00187 this->_c = this->IMPL::copy(o).cap();
00188 return *this;
00189 }
00190 };
00191
00192 }} // L4::Detail

```

## 15.293 l4/sys/cxx/types File Reference

This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Types::Flags< BITS\\_ENUM, UNDERLYING >](#)  
*Template for defining typical [Flags](#) bitmaps.*
- struct [L4::Types::Bool< V >](#)  
*Boolean meta type.*
- struct [L4::Types::False](#)  
*[False](#) meta value.*
- struct [L4::Types::True](#)  
*[True](#) meta value.*
- struct [L4::Types::Same< A, B >](#)  
*Compare two data types for equality.*

## Namespaces

- [L4](#)  
*[L4](#) low-level kernel interface.*
- [L4::Types](#)  
*[L4](#) basic type helpers for C++.*

## 15.294 types

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018
00020
00021 #pragma once
00022
00023 // very simple type traits for basic L4 functions, for a more complete set
00024 // use <l4/cxx/type_traits> or the standard <type_traits>.
00025
00026 namespace L4 {
00027
00031 namespace Types {
00032
00062 template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
00063 class Flags
00064 {
00065 public:
00067 typedef UNDERLYING value_type;
00069 typedef BITS_ENUM bits_enum_type;
00071 typedef Flags<BITS_ENUM, UNDERLYING> type;
00072
00073 private:
00074 struct Private_bool;
00075 value_type _v;
00076 explicit Flags(value_type v) : _v(v) {}
00077
00078 public:
00080 enum None_type { None };
00081
00090 Flags(None_type) : _v(0) {}
00091
00093 Flags() : _v(0) {}
00094
00103 Flags(BITS_ENUM e) : _v(((value_type)1) << e) {}
00104
00110 static type from_raw(value_type v) { return type(v); }
00111
00113 operator Private_bool * () const
00114 { return _v != 0 ? (Private_bool *)1 : 0; }
00115
00117 bool operator ! () const { return _v == 0; }
00118
00120 friend type operator | (type lhs, type rhs)
00121 { return type(lhs._v | rhs._v); }
00122
00124 friend type operator | (type lhs, bits_enum_type rhs)
00125 { return lhs | type(rhs); }
00126
00128 friend type operator & (type lhs, type rhs)
00129 { return type(lhs._v & rhs._v); }
00130
00132 friend type operator & (type lhs, bits_enum_type rhs)
00133 { return lhs & type(rhs); }
00134
00136 type &operator |= (type rhs) { _v |= rhs._v; return *this; }
00138 type &operator |= (bits_enum_type rhs) { return operator |= (
type(rhs)); }
00139
00141 type &operator &= (type rhs) { _v &= rhs._v; return *this; }
00143 type &operator &= (bits_enum_type rhs) { return operator &= (
type(rhs)); }
00144
00146 type operator ~ () const { return type(~_v); }
00147
00154 type &clear(bits_enum_type flag) { return operator &= (~
type(flag)); }
00155
00157 value_type as_value() const { return _v; }
00158 };

```

```

00159
00165 template< bool V > struct Bool
00166 {
00167 typedef Bool<V> type;
00168 enum { value = V };
00169 };
00170
00173 struct False : Bool<false> {};
00174
00177 struct True : Bool<true> {};
00178
00179 /*****/
00188 template<typename A, typename B>
00189 struct Same : False {};
00190
00191 template<typename A>
00192 struct Same<A, A> : True {};
00193
00194 template<bool EXP, typename T = void> struct Enable_if {};
00195 template<typename T> struct Enable_if<true, T> { typedef T type; };
00196
00197 template<typename T1, typename T2, typename T = void>
00198 struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};
00199
00200 template<typename T> struct Remove_const { typedef T type; };
00201 template<typename T> struct Remove_const<T const> { typedef T type; };
00202 template<typename T> struct Remove_volatile { typedef T type; };
00203 template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
00204 template<typename T> struct Remove_cv
00205 { typedef typename Remove_const<typename Remove_volatile<T>::type>::type
type; };
00206
00207 template<typename T> struct Remove_pointer { typedef T type; };
00208 template<typename T> struct Remove_pointer<T*> { typedef T type; };
00209 template<typename T> struct Remove_reference { typedef T type; };
00210 template<typename T> struct Remove_reference<T&> { typedef T type; };
00211 template<typename T> struct Remove_pr { typedef T type; };
00212 template<typename T> struct Remove_pr<T&> { typedef T type; };
00213 template<typename T> struct Remove_pr<T*> { typedef T type; };
00214 } // Types
00215 } // L4

```

## 15.295 l4/sys/debugger File Reference

The debugger interface specifies common debugging related definitions.

```

#include <l4/sys/debugger.h>
#include <l4/sys/kobject>

```





```

00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/debugger.h>
00027 #include <l4/sys/kobject>
00028
00029 namespace L4 {
00030
00053 class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
00054 {
00055 public:
00056 enum
00057 {
00058 Switch_log_on = L4_DEBUGGER_SWITCH_LOG_ON,
00059 Switch_log_off = L4_DEBUGGER_SWITCH_LOG_OFF,
00060 };
00061
00070 l4_msgtag_t set_object_name(const char *name,
00071 l4_utcb_t *utcb = l4_utcb()) throw()
00072 { return l4_debugger_set_object_name_u(cap(), name, utcb); }
00073
00082 unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) throw()
00083 { return l4_debugger_global_id_u(cap(), utcb); }
00084
00094 unsigned long kobj_to_id(l4_addr_t kobjp,
00095 l4_utcb_t *utcb = l4_utcb()) throw()
00096 { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }
00097
00108 int query_log_typeid(const char *name, unsigned idx,
00109 l4_utcb_t *utcb = l4_utcb()) throw()
00110 { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }
00111
00127 int query_log_name(unsigned idx,
00128 char *name, unsigned namelen,
00129 char *shortname, unsigned shortnamelen,
00130 l4_utcb_t *utcb = l4_utcb()) throw()
00131 {
00132 return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
00133 shortname, shortnamelen, utcb);
00134 }
00135
00144 l4_msgtag_t switch_log(const char *name, unsigned on_off,
00145 l4_utcb_t *utcb = l4_utcb()) throw()
00146 { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }
00147
00159 l4_msgtag_t get_object_name(unsigned id, char *name, unsigned size,
00160 l4_utcb_t *utcb = l4_utcb()) throw()
00161 { return l4_debugger_get_object_name_u(cap(), id, name, size, utcb); }
00162 };
00163 }

```

## 15.297 l4/sys/debugger.h File Reference

Debugger related definitions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/kernel_object.h>

```



*Get the globally unique ID of the object behind the kobject pointer.*

- int [l4\\_debugger\\_query\\_log\\_typeid](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, unsigned idx) [L4\\_NOTHROW](#)

*Query the log-id for a log type.*

- int [l4\\_debugger\\_query\\_log\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned idx, char \*name, unsigned namelen, char \*shortname, unsigned shortnamelen) [L4\\_NOTHROW](#)

*Query the name of a log type given the ID.*

- [l4\\_msgtag\\_t](#) [l4\\_debugger\\_switch\\_log](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, int on\_off) [L4\\_NOTHROW](#)

*Set or unset log.*

## 15.297.1 Detailed Description

Debugger related definitions.

Definition in file [debugger.h](#).

## 15.297.2 Function Documentation

### 15.297.2.1 l4\_debugger\_get\_object\_name()

```
l4_msgtag_t l4_debugger_get_object_name (
 l4_cap_idx_t cap,
 unsigned id,
 char * name,
 unsigned size) [inline]
```

Get name of the kernel object with Id id.

#### Parameters

|     |             |                                                                           |
|-----|-------------|---------------------------------------------------------------------------|
|     | <i>cap</i>  | Capability of the debugger object.                                        |
|     | <i>id</i>   | Global id of the object whose name is asked.                              |
| out | <i>name</i> | Buffer to copy the name into. The buffer must be allocated by the caller. |
|     | <i>size</i> | Length of the <i>name</i> buffer.                                         |

#### Returns

Syscall return tag

Definition at line 366 of file [debugger.h](#).

### 15.297.2.2 l4\_debugger\_query\_log\_name()

```
int l4_debugger_query_log_name (
 l4_cap_idx_t cap,
```

```
unsigned idx,
char * name,
unsigned namelen,
char * shortname,
unsigned shortnamelen) [inline]
```

Query the name of a log type given the ID.

#### Parameters

|                     |                              |
|---------------------|------------------------------|
| <i>cap</i>          | Debugger capability.         |
| <i>idx</i>          | ID to query.                 |
| <i>name</i>         | Buffer to copy name to.      |
| <i>namelen</i>      | Buffer length of name.       |
| <i>shortname</i>    | Buffer to copy shortname to. |
| <i>shortnamelen</i> | Buffer length of shortname.  |

#### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

This is a debugging facility, the call might be invalid.

Definition at line 350 of file [debugger.h](#).

#### 15.297.2.3 l4\_debugger\_query\_log\_typeid()

```
int l4_debugger_query_log_typeid (
 l4_cap_idx_t cap,
 const char * name,
 unsigned idx) [inline]
```

Query the log-id for a log type.

#### Parameters

|             |                                      |
|-------------|--------------------------------------|
| <i>cap</i>  | Debugger capability                  |
| <i>name</i> | Name to query for.                   |
| <i>idx</i>  | Idx to start searching, start with 0 |

#### Returns

positive ID, or negative error code

This is a debugging facility, the call might be invalid.

Definition at line 343 of file [debugger.h](#).

## 15.297.2.4 l4\_debugger\_switch\_log()

```
l4_msgtag_t l4_debugger_switch_log (
 l4_cap_idx_t cap,
 const char * name,
 int on_off) [inline]
```

Set or unset log.

## Parameters

|               |                                 |
|---------------|---------------------------------|
| <i>cap</i>    | Debugger object.                |
| <i>name</i>   | Name of the log type.           |
| <i>on_off</i> | 1: turn log on, 0: turn log off |

## Returns

Syscall return tag

Definition at line 359 of file [debugger.h](#).

## 15.298 debugger.h

```
00001 #pragma once
00002
00007 /*
00008 * (c) 2008-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/utcb.h>
00028 #include <l4/sys/ipc.h>
00029
00053 L4_INLINE l4_msgtag_t
00054 l4_debugger_set_object_name(l4_cap_idx_t cap, const char *name)
00055 L4_NOTHROW;
00056
00059 L4_INLINE l4_msgtag_t
00060 l4_debugger_set_object_name_u(l4_cap_idx_t cap, const char *name,
00061 l4_utcb_t *utcb) L4_NOTHROW;
00062
00073 L4_INLINE l4_msgtag_t
00074 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00075 char *name, unsigned size) L4_NOTHROW;
00076
00080 L4_INLINE l4_msgtag_t
00081 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00082 char *name, unsigned size,
00083 l4_utcb_t *utcb) L4_NOTHROW;
00084
00096 L4_INLINE unsigned long
00097 l4_debugger_global_id(l4_cap_idx_t cap)
00098 L4_NOTHROW;
```

```

00098
00102 L4_INLINE unsigned long
00103 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb)
00104 L4_NOTHROW;
00104
00117 L4_INLINE unsigned long
00118 l4_debugger_kobj_to_id(l4_cap_idx_t cap,
00119 l4_addr_t kobjp) L4_NOTHROW;
00119
00123 L4_INLINE unsigned long
00124 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp,
00125 l4_utcb_t *utcb) L4_NOTHROW;
00125
00137 L4_INLINE int
00138 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00139 unsigned idx) L4_NOTHROW;
00140
00144 L4_INLINE int
00145 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00146 unsigned idx, l4_utcb_t *utcb)
00147 L4_NOTHROW;
00147
00163 L4_INLINE int
00164 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00165 char *name, unsigned namelen,
00166 char *shortname, unsigned shortnamelen) L4_NOTHROW;
00167
00171 L4_INLINE int
00172 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00173 char *name, unsigned namelen,
00174 char *shortname, unsigned shortnamelen,
00175 l4_utcb_t *utcb) L4_NOTHROW;
00176
00186 L4_INLINE l4_msgtag_t
00187 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00188 int on_off) L4_NOTHROW;
00189
00193 L4_INLINE l4_msgtag_t
00194 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00195 l4_utcb_t *utcb) L4_NOTHROW;
00196
00197 enum
00198 {
00199 L4_DEBUGGER_NAME_SET_OP = 0UL,
00200 L4_DEBUGGER_GLOBAL_ID_OP = 1UL,
00201 L4_DEBUGGER_KOBJ_TO_ID_OP = 2UL,
00202 L4_DEBUGGER_QUERY_LOG_TYPEID_OP = 3UL,
00203 L4_DEBUGGER_SWITCH_LOG_OP = 4UL,
00204 L4_DEBUGGER_NAME_GET_OP = 5UL,
00205 L4_DEBUGGER_QUERY_LOG_NAME_OP = 6UL,
00206 };
00207
00208 enum
00209 {
00210 L4_DEBUGGER_SWITCH_LOG_ON = 1,
00211 L4_DEBUGGER_SWITCH_LOG_OFF = 0,
00212 };
00213
00214 /* IMPLEMENTATION -----*/
00215
00216 #include <l4/sys/kernel_object.h>
00217
00218 L4_INLINE l4_msgtag_t
00219 l4_debugger_set_object_name_u(unsigned long cap,
00220 const char *name, l4_utcb_t *utcb)
00221 L4_NOTHROW
00222 {
00223 unsigned int i;
00224 char *s = (char *)&l4_utcb_mr_u(utcb)->mr[1];
00225 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
00226 for (i = 0;
00227 *name && i < (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(
00228 l4_umword_t) - 1;
00229 ++i, ++name, ++s)
00230 *s = *name;
00231 *s = 0;
00232 i = (i + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00233 return l4_invoke_debugger(cap, l4_msgtag(0, i + 1, 0, 0), utcb);
00234 }
00234
00234 L4_INLINE unsigned long
00235 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb)
00236 L4_NOTHROW
00237 {
00238 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_OP;
00239 if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
00240 return ~0UL;

```

```

00240 return l4_utcb_mr_u(utcb)->mr[0];
00241 }
00242
00243 L4_INLINE unsigned long
00244 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp,
00245 l4_utcb_t *utcb) L4_NOTHROW
00246 {
00247 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_TO_ID_OP;
00248 l4_utcb_mr_u(utcb)->mr[1] = kobjp;
00249 if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
00250 return ~0UL;
00251 return l4_utcb_mr_u(utcb)->mr[0];
00252 }
00253 L4_INLINE int
00254 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00255 unsigned idx,
00256 l4_utcb_t *utcb) L4_NOTHROW
00257 {
00258 unsigned l;
00259 int e;
00260 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
00261 l4_utcb_mr_u(utcb)->mr[1] = idx;
00262 l = __builtin_strlen(name);
00263 l = l > 31 ? 31 : l;
00264 __builtin_strncpy((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 31);
00265 l = (l + 1 + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00266 e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + l, 0, 0), utcb), utcb);
00267 if (e < 0)
00268 return e;
00269 return l4_utcb_mr_u(utcb)->mr[0];
00270 }
00271
00272 L4_INLINE int
00273 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00274 char *name, unsigned namelen,
00275 char *shortname, unsigned shortnamelen,
00276 l4_utcb_t *utcb) L4_NOTHROW
00277 {
00278 int e;
00279 char *n;
00280 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_NAME_OP;
00281 l4_utcb_mr_u(utcb)->mr[1] = idx;
00282 e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
00283 if (e < 0)
00284 return e;
00285 n = (char *)&l4_utcb_mr_u(utcb)->mr[0];
00286 __builtin_strncpy(name, n, namelen);
00287 name[namelen - 1] = 0;
00288 __builtin_strncpy(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
00289 shortname[shortnamelen - 1] = 0;
00290 return 0;
00291 }
00292
00293
00294 L4_INLINE l4_msgtag_t
00295 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00296 l4_utcb_t *utcb) L4_NOTHROW
00297 {
00298 unsigned l;
00299 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
00300 l4_utcb_mr_u(utcb)->mr[1] = on_off;
00301 l = __builtin_strlen(name);
00302 l = l > 31 ? 31 : l;
00303 __builtin_strncpy((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 31);
00304 l = (l + 1 + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00305 return l4_invoke_debugger(cap, l4_msgtag(0, 2 + l, 0, 0), utcb);
00306 }
00307
00308 L4_INLINE l4_msgtag_t
00309 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00310 char *name, unsigned size,
00311 l4_utcb_t *utcb) L4_NOTHROW
00312 {
00313 l4_msgtag_t t;
00314 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_GET_OP;
00315 l4_utcb_mr_u(utcb)->mr[1] = id;
00316 t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
00317 __builtin_strncpy(name, (char *)&l4_utcb_mr_u(utcb)->mr[0], size);
00318 name[size - 1] = 0;
00319 return t;
00320 }
00321
00322
00323 L4_INLINE l4_msgtag_t
00324 l4_debugger_set_object_name(unsigned long cap,
00325 const char *name) L4_NOTHROW

```

```

00326 {
00327 return l4_debugger_set_object_name_u(cap, name, l4_utcb());
00328 }
00329
00330 L4_INLINE unsigned long
00331 l4_debugger_global_id(l4_cap_idx_t cap)
00332 L4_NOTHROW
00333 {
00334 return l4_debugger_global_id_u(cap, l4_utcb());
00335 }
00336 L4_INLINE unsigned long
00337 l4_debugger_kobj_to_id(l4_cap_idx_t cap,
00338 l4_addr_t kobjp) L4_NOTHROW
00339 {
00340 return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
00341 }
00342 L4_INLINE int
00343 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00344 unsigned idx) L4_NOTHROW
00345 {
00346 return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
00347 }
00348
00349 L4_INLINE int
00350 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00351 char *name, unsigned namelen,
00352 char *shortname, unsigned shortnamelen) L4_NOTHROW
00353 {
00354 return l4_debugger_query_log_name_u(cap, idx, name, namelen,
00355 shortname, shortnamelen, l4_utcb());
00356 }
00357
00358 L4_INLINE l4_msgtag_t
00359 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00360 int on_off) L4_NOTHROW
00361 {
00362 return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());
00363 }
00364
00365 L4_INLINE l4_msgtag_t
00366 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00367 char *name, unsigned size) L4_NOTHROW
00368 {
00369 return l4_debugger_get_object_name_u(cap, id, name, size, l4_utcb());
00370 }

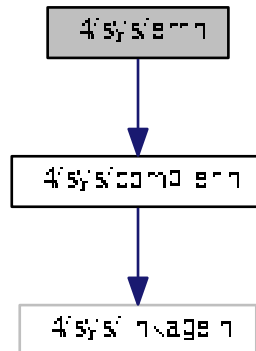
```

## 15.299 l4/sys/err.h File Reference

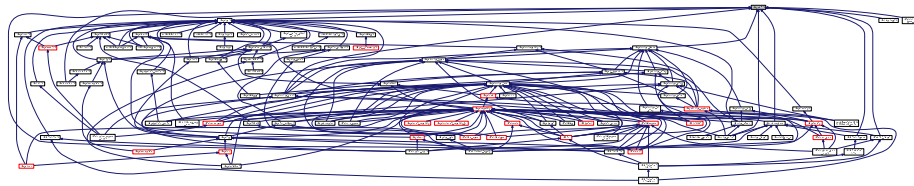
Error codes.



```
#include <l4/sys/compiler.h>
Include dependency graph for err.h:
```



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `l4_error_code_t` {  
`L4_EOK` = 0, `L4_EPERM` = 1, `L4_ENOENT` = 2, `L4_EIO` = 5,  
`L4_ENXIO` = 6, `L4_E2BIG` = 7, `L4_EAGAIN` = 11, `L4_ENOMEM` = 12,  
`L4_EACCESS` = 13, `L4_EFAULT` = 14, `L4_EBUSY` = 16, `L4_EEXIST` = 17,  
`L4_ENODEV` = 19, `L4_EINVAL` = 22, `L4_ENOSPC` = 28, `L4_ERANGE` = 34,  
`L4_ENAMETOOLONG` = 36, `L4_ENOSYS` = 38, `L4_EBADPROTO` = 39, `L4_EADDRNOTAVAIL` = 99,  
`L4_ERRNOMAX` = 100, `L4_ENOREPLY` = 1000, `L4_MSGTOOSHORT` = 1001, `L4_MSGTOOLONG` =  
1002,  
`L4_MSGMISSARG` = 1003, `L4_EIPC_LO` = 2000, `L4_EIPC_HI` = 2000 + 0x1f }

*L4 error codes.*

### 15.299.1 Detailed Description

Error codes.

Definition in file `err.h`.

## 15.300 err.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00041 enum l4_error_code_t
00042 {
00043 L4_EOK = 0,
00044 L4_EPERM = 1,
00045 L4_ENOENT = 2,
00046 L4_EIO = 5,
00047 L4_ENXIO = 6,
00048 L4_E2BIG = 7,
00049 L4_EAGAIN = 11,
00050 L4_ENOMEM = 12,
00051 L4_EACCESS = 13,
00052 L4_EFAULT = 14,
00053 L4_EBUSY = 16,
00054 L4_EEXIST = 17,
00055 L4_ENODEV = 19,
00056 L4_EINVAL = 22,
00057 L4_ENOSPC = 28,
00058 L4_ERANGE = 34,
00059 L4_ENAMETOOLONG = 36,
00060 L4_ENOSYS = 38,
00061 L4_EBADPROTO = 39,
00062 L4_EADDRNOTAVAIL = 99,
00063 L4_ERRNOMAX = 100,
00065 L4_ENOREPLY = 1000,
00066 L4_MSGTOOSHORT = 1001,
00067 L4_MSGTOOLONG = 1002,
00068 L4_MSGMISSARG = 1003,
00070 L4_EIPC_LO = 2000,
00071 L4_EIPC_HI = 2000 + 0x1f,
00072 };
00073
00074 __BEGIN_DECLS
00075 L4_CV char const *l4sys_errtostr(long err) L4_NOTHROW;
00076 __END_DECLS
00077
00078

```

## 15.301 l4/sys/exception File Reference

Exception C++ interface.

```

#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```

[illegible]

- class L4::Exception  
*Exception interface.*

- L4  
*L4 low-level kernel interface.*

Definition in file [exception](#).

## 15.302 exception

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/cxx/ipc_types>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4 {
00031
00042 class L4_EXPORT Exception :
00043 public Kobject_0t<Exception, L4_PROTO_EXCEPTION>
00044 {
00045 public:
00046 // TODO: pass a reference/pointer to the UTCB not copy the regs
00047 L4_INLINE_RPC(
00048 l4_msgtag_t, exception, (L4::Ipc::In_out<l4_exc_regs_t *>
00049 regs,
00050 L4::Ipc::Rcv_fpage rwin,
00051 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00052
00062 typedef L4::Typeid::Rpc_nocode<exception_t>
00063 Rpcs;
00064 };
00065 }

```

## 15.303 l4/sys/factory File Reference

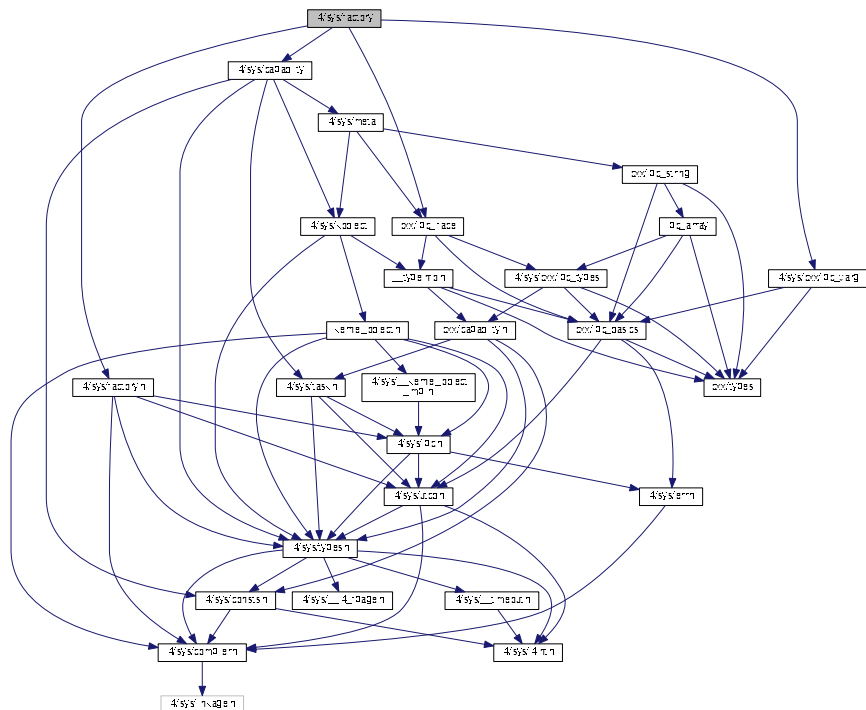
Common factory related definitions.

```

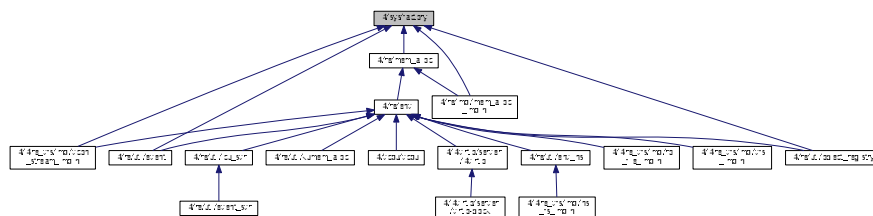
#include <l4/sys/factory.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_varg>

```

Include dependency graph for factory:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Factory](#)  
*C++ [Factory](#) interface to create kernel objects.*
- struct [L4::Factory::Nil](#)  
*Special type to add a void argument into the factory create stream.*
- struct [L4::Factory::Lstr](#)  
*Special type to add a pascal string into the factory create stream.*
- class [L4::Factory::S](#)  
*Stream class for the [create\(\)](#) argument stream.*

## Namespaces

- [L4](#)  
*[L4](#) low-level kernel interface.*

### 15.303.1 Detailed Description

Common factory related definitions.

Definition in file [factory](#).

## 15.304 factory

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021 #include <l4/sys/factory.h>
00022 #include <l4/sys/capability>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/cxx/ipc_vararg>
00025 namespace L4 {
00026 class Factory : public Kobject_t<Factory, Kobject, L4_PROTO_FACTORY>
00027 {
00028 public:
00029 typedef l4_mword_t Proto;
00030 struct Nil {};
00031 struct Lstr
00032 {
00033 char const *s;
00034 int len;
00035 Lstr(char const *s, int len) : s(s), len(len) {}
00036 };
00037 class S
00038 {
00039 private:
00040 l4_utcb_t *u;
00041 l4_msgtag_t t;
00042 l4_cap_idx_t f;
00043 public:
00044 S(S const &o)
00045 : u(o.u), t(o.t), f(o.f)
00046 { const_cast<S&>(o).t.raw = 0; }
00047 S(l4_cap_idx_t f, long obj, L4::Cap<void> target,
00048 l4_utcb_t *utcb) throw()
00049 : u(utcb), t(l4_factory_create_start_u(obj, target.cap(), u)), f(f)
00050 {}
00051 ~S()
00052 {
00053 if (t.raw)
00054 l4_factory_create_commit_u(f, t, u);
00055 }
00056 operator l4_msgtag_t ()
00057 {

```

```

00146 l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
00147 t.raw = 0;
00148 return r;
00149 }
00150
00158 S &operator << (l4_mword_t i)
00159 {
00160 l4_factory_create_add_int_u(i, &t, u);
00161 return *this;
00162 }
00163
00171 S &operator << (l4_umword_t i)
00172 {
00173 l4_factory_create_add_uint_u(i, &t, u);
00174 return *this;
00175 }
00176
00186 S &operator << (char const *s)
00187 {
00188 l4_factory_create_add_str_u(s, &t, u);
00189 return *this;
00190 }
00191
00203 S &operator << (Lstr const &s)
00204 {
00205 l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
00206 return *this;
00207 }
00208
00214 S &operator << (Nil)
00215 {
00216 l4_factory_create_add_nil_u(&t, u);
00217 return *this;
00218 }
00219
00227 S &operator << (l4_fpage_t d)
00228 {
00229 l4_factory_create_add_fpage_u(d, &t, u);
00230 return *this;
00231 }
00232 };
00233
00234
00235 public:
00236
00259 S create(Cap<void> target, long obj, l4_utcb_t *utcb =
l4_utcb()) throw()
00260 {
00261 return S(cap(), obj, target, utcb);
00262 }
00263
00275 template<typename OBJ>
00276 S create(Cap<OBJ> target, l4_utcb_t *utcb = l4_utcb()) throw()
00277 {
00278 return S(cap(), OBJ::Protocol, target, utcb);
00279 }
00280
00281 L4_INLINE_RPC_NF(
00282 l4_msgtag_t, create, (L4::Ipc::Out<
L4::Cap<void> > target, l4_mword_t obj,
00283 L4::Ipc::Varg const *args),
00284 L4::Ipc::Call_t<L4_CAP_FPAGE_S>);
00285
00306 l4_msgtag_t create_task(Cap<Task> const & target_cap,
00307 l4_fpage_t const &utcb_area,
00308 l4_utcb_t *utcb = l4_utcb()) throw()
00309 { return l4_factory_create_task_u(cap(), target_cap.
cap(), utcb_area, utcb); }
00310
00324 l4_msgtag_t create_thread(Cap<Thread> const &target_cap,
00325 l4_utcb_t *utcb = l4_utcb()) throw()
00326 { L4_DEPRECATED("Call create with Cap<Thread> as argument instead.")
00327 { return l4_factory_create_thread_u(cap(), target_cap.
cap(), utcb); } }
00328
00342 l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
00343 unsigned long limit,
00344 l4_utcb_t *utcb = l4_utcb()) throw()
00345 { return l4_factory_create_factory_u(cap(), target_cap.
cap(), limit, utcb); }
00346
00372 l4_msgtag_t create_gate(Cap<void> const &target_cap,
00373 Cap<Thread> const &thread_cap, l4_umword_t label,
00374 l4_utcb_t *utcb = l4_utcb()) throw()
00375 { return l4_factory_create_gate_u(cap(), target_cap.
cap(), thread_cap.cap(), label, utcb); }
00376

```

```

00390 l4_msgtag_t create_irq(Cap<Irq>const &target_cap,
00391 l4_utcb_t *utcb = l4_utcb()) throw()
00392 L4_DEPRECATED("Call create with Cap<Irq> as argument instead.")
00393 { return l4_factory_create_irq_u(cap(), target_cap,
00394 cap(), utcb); }
00394
00408 l4_msgtag_t create_vm(Cap<Vm>const &target_cap,
00409 l4_utcb_t *utcb = l4_utcb()) throw()
00410 L4_DEPRECATED("Call create with Cap<Vm> as argument instead.")
00411 { return l4_factory_create_vm_u(cap(), target_cap.cap(), utcb); }
00412
00413 typedef L4::Typeid::Rpc_nocode<create_t> Rpcs;
00414 };
00415
00416 }

```

## 15.305 l4/sys/factory.h File Reference

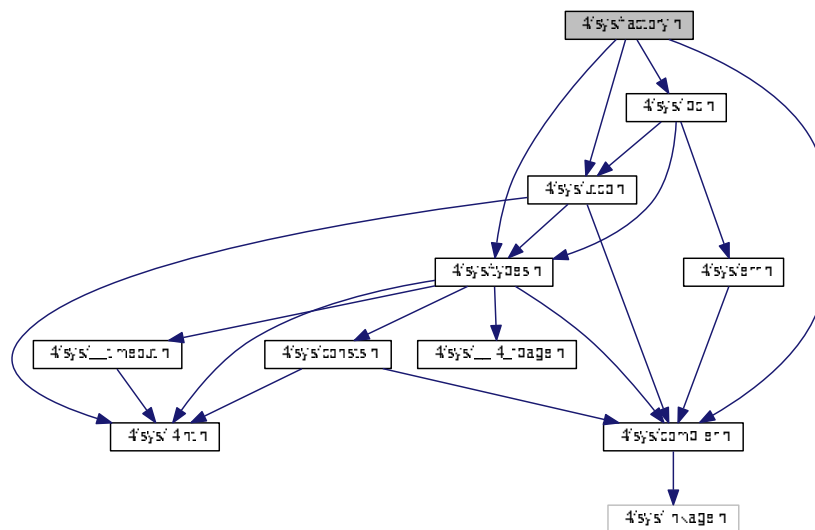
Common factory related definitions.

```

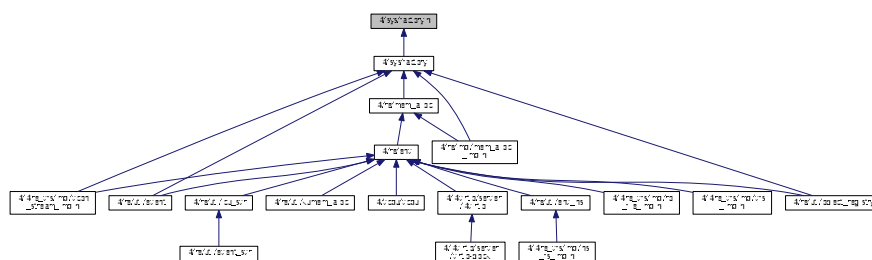
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for factory.h:



This graph shows which files directly or indirectly include this file:





## Functions

- `l4_msgtag_t l4_factory_create_task (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t const utcb_area) L4_NOTHROW`  
*Create a new task.*
- `l4_msgtag_t l4_factory_create_task_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t const utcb_area, l4_utcb_t *utcb) L4_NOTHROW`  
*Create a new task.*
- `l4_msgtag_t l4_factory_create_thread (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`  
*Create a new thread.*
- `l4_msgtag_t l4_factory_create_thread_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`  
*Create a new thread.*
- `l4_msgtag_t l4_factory_create_factory (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW`  
*Create a new factory.*
- `l4_msgtag_t l4_factory_create_factory_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW`  
*Create a new factory.*
- `l4_msgtag_t l4_factory_create_gate (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label) L4_NOTHROW`  
*Create a new IPC gate.*
- `l4_msgtag_t l4_factory_create_gate_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label, l4_utcb_t *utcb) L4_NOTHROW`  
*Create a new IPC gate.*
- `l4_msgtag_t l4_factory_create_irq (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`  
*Create a new IRQ sender.*
- `l4_msgtag_t l4_factory_create_irq_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`  
*Create a new IRQ.*
- `l4_msgtag_t l4_factory_create_vm (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`  
*Create a new virtual machine.*
- `l4_msgtag_t l4_factory_create_vm_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`  
*Create a new virtual machine.*

### 15.305.1 Detailed Description

Common factory related definitions.

Definition in file [factory.h](#).

## 15.306 factory.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>,
00011 * Henning Schild <hschild@os.inf.tu-dresden.de>
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
```

```

00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 #pragma once
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/utcb.h>
00032
00086 L4_INLINE l4_msgtag_t
00087 l4_factory_create_task(l4_cap_idx_t factory,
00088 l4_cap_idx_t target_cap, l4_fpage_t const utcb_area)
00089 L4_NOTHROW;
00096 L4_INLINE l4_msgtag_t
00097 l4_factory_create_task_u(l4_cap_idx_t factory,
00098 l4_cap_idx_t target_cap,
00099 l4_fpage_t const utcb_area, l4_utcb_t *utcb)
00100 L4_NOTHROW;
00111 L4_INLINE l4_msgtag_t
00112 l4_factory_create_thread(l4_cap_idx_t factory,
00113 l4_cap_idx_t target_cap) L4_NOTHROW;
00114
00121 L4_INLINE l4_msgtag_t
00122 l4_factory_create_thread_u(l4_cap_idx_t factory,
00123 l4_cap_idx_t target_cap, l4_utcb_t *utcb)
00124 L4_NOTHROW;
00138 L4_INLINE l4_msgtag_t
00139 l4_factory_create_factory(l4_cap_idx_t factory,
00140 l4_cap_idx_t target_cap,
00141 unsigned long limit) L4_NOTHROW;
00148 L4_INLINE l4_msgtag_t
00149 l4_factory_create_factory_u(l4_cap_idx_t factory,
00150 l4_cap_idx_t target_cap,
00151 unsigned long limit, l4_utcb_t *utcb)
00152 L4_NOTHROW;
00176 L4_INLINE l4_msgtag_t
00177 l4_factory_create_gate(l4_cap_idx_t factory,
00178 l4_cap_idx_t target_cap,
00179 l4_cap_idx_t thread_cap, l4_umword_t label)
00180 L4_NOTHROW;
00187 L4_INLINE l4_msgtag_t
00188 l4_factory_create_gate_u(l4_cap_idx_t factory,
00189 l4_cap_idx_t target_cap,
00190 l4_cap_idx_t thread_cap, l4_umword_t label,
00191 l4_utcb_t *utcb) L4_NOTHROW;
00192
00205 L4_INLINE l4_msgtag_t
00206 l4_factory_create_irq(l4_cap_idx_t factory,
00207 l4_cap_idx_t target_cap) L4_NOTHROW;
00208
00215 L4_INLINE l4_msgtag_t
00216 l4_factory_create_irq_u(l4_cap_idx_t factory,
00217 l4_cap_idx_t target_cap, l4_utcb_t *utcb)
00218 L4_NOTHROW;
00230 L4_INLINE l4_msgtag_t
00231 l4_factory_create_vm(l4_cap_idx_t factory,
00232 l4_cap_idx_t target_cap) L4_NOTHROW;
00233
00240 L4_INLINE l4_msgtag_t
00241 l4_factory_create_vm_u(l4_cap_idx_t factory,
00242 l4_cap_idx_t target_cap, l4_utcb_t *utcb)
00243 L4_NOTHROW;
00244 L4_INLINE l4_msgtag_t
00245 l4_factory_create_start_u(long obj, l4_cap_idx_t target,
00246 l4_utcb_t *utcb) L4_NOTHROW;
00247
00248 L4_INLINE int
00249 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00250 l4_utcb_t *utcb) L4_NOTHROW;

```

```

00251
00252 L4_INLINE int
00253 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00254 l4_utcb_t *utcb) L4_NOTHROW;
00255
00256 L4_INLINE int
00257 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00258 l4_utcb_t *utcb) L4_NOTHROW;
00259
00260 L4_INLINE int
00261 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00262 l4_utcb_t *utcb) L4_NOTHROW;
00263
00264 L4_INLINE int
00265 l4_factory_create_add_lstr_u(char const *s, int len, l4_msgtag_t *tag,
00266 l4_utcb_t *utcb) L4_NOTHROW;
00267
00268 L4_INLINE int
00269 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb)
00270 L4_NOTHROW;
00271
00272 L4_INLINE l4_msgtag_t
00273 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00274 l4_utcb_t *utcb) L4_NOTHROW;
00275
00276 L4_INLINE l4_msgtag_t
00277 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00278 l4_utcb_t *utcb) L4_NOTHROW;
00279
00280 L4_INLINE l4_msgtag_t
00281 l4_factory_create(l4_cap_idx_t factory, long obj,
00282 l4_cap_idx_t target) L4_NOTHROW;
00283
00284 /* IMPLEMENTATION -----*/
00285
00286 #include <l4/sys/ipc.h>
00287
00288 L4_INLINE l4_msgtag_t
00289 l4_factory_create_task_u(l4_cap_idx_t factory,
00290 l4_cap_idx_t target_cap, l4_fpage_t utcb_area,
00291 l4_utcb_t *u) L4_NOTHROW
00292 {
00293 l4_msgtag_t t;
00294 t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
00295 l4_factory_create_add_fpage_u(utcb_area, &t, u);
00296 return l4_factory_create_commit_u(factory, t, u);
00297 }
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_factory_create_thread_u(l4_cap_idx_t factory,
00301 l4_cap_idx_t target_cap, l4_utcb_t *u)
00302 L4_NOTHROW
00303 {
00304 return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
00305 }
00306
00307 L4_INLINE l4_msgtag_t
00308 l4_factory_create_factory_u(l4_cap_idx_t factory,
00309 l4_cap_idx_t target_cap, unsigned long limit,
00310 l4_utcb_t *u) L4_NOTHROW
00311 {
00312 l4_msgtag_t t;
00313 t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
00314 l4_factory_create_add_uint_u(limit, &t, u);
00315 return l4_factory_create_commit_u(factory, t, u);
00316 }
00317
00318 L4_INLINE l4_msgtag_t
00319 l4_factory_create_gate_u(l4_cap_idx_t factory,
00320 l4_cap_idx_t target_cap,
00321 l4_cap_idx_t thread_cap, l4_umword_t label,
00322 l4_utcb_t *u) L4_NOTHROW
00323 {
00324 l4_msgtag_t t;
00325 l4_msg_regs_t *v;
00326 int items = 0;
00327 t = l4_factory_create_start_u(0, target_cap, u);
00328 l4_factory_create_add_uint_u(label, &t, u);
00329 v = l4_utcb_mr_u(u);
00330 if (!(thread_cap & L4_INVALID_CAP_BIT))
00331 {
00332 items = 1;
00333 v->mr[3] = l4_map_obj_control(0,0);
00334 v->mr[4] = l4_obj_fpage(thread_cap, 0, L4_FPAGE_RWX).
00335 raw;
00336 }
00337 return l4_factory_create_commit_u(factory, t, u);
00338 }

```

```

00335 t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items,
00336 l4_msgtag_flags(t));
00337 return l4_factory_create_commit_u(factory, t, u);
00338 }
00339 L4_INLINE l4_msgtag_t
00340 l4_factory_create_irq_u(l4_cap_idx_t factory,
00341 l4_cap_idx_t target_cap, l4_utcb_t *u)
00342 {
00343 L4_NOTHROW
00344 return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
00345 }
00346 L4_INLINE l4_msgtag_t
00347 l4_factory_create_vm_u(l4_cap_idx_t factory,
00348 l4_cap_idx_t target_cap,
00349 l4_utcb_t *u) L4_NOTHROW
00350 {
00351 return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
00352 }
00353
00354
00355
00356
00357
00358 L4_INLINE l4_msgtag_t
00359 l4_factory_create_task(l4_cap_idx_t factory,
00360 l4_cap_idx_t target_cap, l4_fpage_t const utcb_area)
00361 {
00362 L4_NOTHROW
00363 return l4_factory_create_task_u(factory, target_cap, utcb_area,
00364 l4_utcb());
00365 }
00366 L4_INLINE l4_msgtag_t
00367 l4_factory_create_thread(l4_cap_idx_t factory,
00368 l4_cap_idx_t target_cap) L4_NOTHROW
00369 {
00370 return l4_factory_create_thread_u(factory, target_cap,
00371 l4_utcb());
00372 }
00373 L4_INLINE l4_msgtag_t
00374 l4_factory_create_factory(l4_cap_idx_t factory,
00375 l4_cap_idx_t target_cap, unsigned long limit)
00376 {
00377 L4_NOTHROW
00378 return l4_factory_create_factory_u(factory, target_cap, limit,
00379 l4_utcb());
00380 }
00381 L4_INLINE l4_msgtag_t
00382 l4_factory_create_gate(l4_cap_idx_t factory,
00383 l4_cap_idx_t target_cap,
00384 l4_cap_idx_t thread_cap, l4_umword_t label)
00385 {
00386 L4_NOTHROW
00387 return l4_factory_create_gate_u(factory, target_cap, thread_cap, label,
00388 l4_utcb());
00389 }
00390 L4_INLINE l4_msgtag_t
00391 l4_factory_create_irq(l4_cap_idx_t factory,
00392 l4_cap_idx_t target_cap) L4_NOTHROW
00393 {
00394 return l4_factory_create_irq_u(factory, target_cap,
00395 l4_utcb());
00396 }
00397 L4_INLINE l4_msgtag_t
00398 l4_factory_create_vm(l4_cap_idx_t factory,
00399 l4_cap_idx_t target_cap) L4_NOTHROW
00400 {
00401 return l4_factory_create_vm_u(factory, target_cap,
00402 l4_utcb());
00403 }
00404 L4_INLINE l4_msgtag_t
00405 l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
00406 l4_utcb_t *u) L4_NOTHROW
00407 {
00408 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00409 l4_buf_regs_t *b = l4_utcb_br_u(u);
00410 v->mr[0] = obj;
00411 b->bdr = 0;
00412 b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;

```

```

00411 return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
00412 }
00413
00414 L4_INLINE int
00415 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00416 l4_utcb_t *u) L4_NOTHROW
00417 {
00418 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00419 int w = l4_msgtag_words(*tag);
00420 if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00421 return 0;
00422 v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) << 16);
00423 v->mr[w + 1] = d.raw;
00424 w += 2;
00425 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00426 return 1;
00427 }
00428
00429 L4_INLINE int
00430 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00431 l4_utcb_t *u) L4_NOTHROW
00432 {
00433 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00434 int w = l4_msgtag_words(*tag);
00435 if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00436 return 0;
00437 v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) << 16);
00438 v->mr[w + 1] = d;
00439 w += 2;
00440 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00441 return 1;
00442 }
00443
00444 L4_INLINE int
00445 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00446 l4_utcb_t *u) L4_NOTHROW
00447 {
00448 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00449 int w = l4_msgtag_words(*tag);
00450 if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00451 return 0;
00452 v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) << 16);
00453 v->mr[w + 1] = d;
00454 w += 2;
00455 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00456 return 1;
00457 }
00458
00459 L4_INLINE int
00460 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00461 l4_utcb_t *u) L4_NOTHROW
00462 {
00463 return l4_factory_create_add_lstr_u(s, __builtin_strlen(s) + 1, tag, u);
00464 }
00465
00466 L4_INLINE int
00467 l4_factory_create_add_lstr_u(char const *s, int len, l4_msgtag_t *tag,
00468 l4_utcb_t *u) L4_NOTHROW
00469 {
00470 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00471 int w = l4_msgtag_words(*tag);
00472 char *c;
00473 int i;
00474
00475 if (w + 1 + (len + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t)
00476 > L4_UTCB_GENERIC_DATA_SIZE)
00477 return 0;
00478
00479 v->mr[w] = L4_VARG_TYPE_STRING | (len << 16);
00480 c = (char*)&v->mr[w + 1];
00481 for (i = 0; i < len; ++i)
00482 *c++ = *s++;
00483
00484 w = w + 1 + (len + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00485
00486 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00487 return 1;
00488 }
00489
00490
00491 L4_INLINE int
00492 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb)
00493 L4_NOTHROW
00494 {
00495 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00496 int w = l4_msgtag_words(*tag);
00497 v->mr[w] = L4_VARG_TYPE_NIL;

```

```

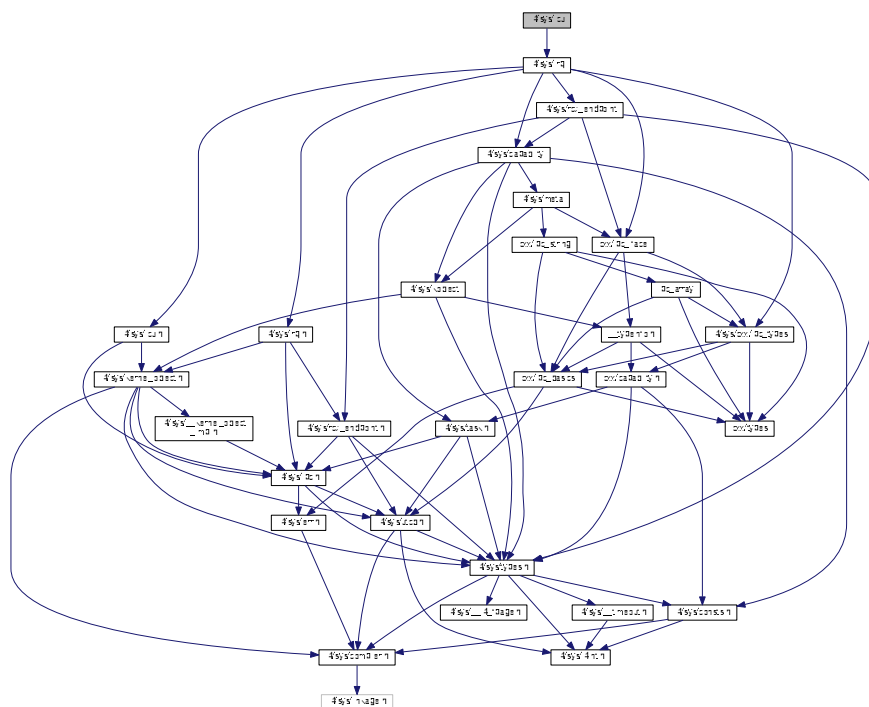
00497 ++w;
00498 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00499 return 1;
00500 }
00501
00502
00503 L4_INLINE l4_msgtag_t
00504 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00505 l4_utcb_t *u) L4_NOTHROW
00506 {
00507 return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
00508 }
00509
00510 L4_INLINE l4_msgtag_t
00511 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00512 l4_utcb_t *utcb) L4_NOTHROW
00513 {
00514 l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
00515 return l4_factory_create_commit_u(factory, t, utcb);
00516 }
00517
00518
00519 L4_INLINE l4_msgtag_t
00520 l4_factory_create(l4_cap_idx_t factory, long obj,
00521 l4_cap_idx_t target) L4_NOTHROW
00522 {
00523 return l4_factory_create_u(factory, obj, target, l4_utcb());
00524 }

```

## 15.307 l4/sys/icu File Reference

Interrupt controller.

```
#include <l4/sys/irq>
Include dependency graph for icu:
```









## Enumerations

- enum `L4_icu_flags` { `L4_ICU_FLAG_MSI` }  
*Flags for IRQ numbers used for the ICU.*
- enum `L4_irq_mode` {  
`L4_IRQ_F_NONE` = 0, `L4_IRQ_F_LEVEL` = 0x2, `L4_IRQ_F_EDGE` = 0x0, `L4_IRQ_F_POS` = 0x0,  
`L4_IRQ_F_NEG` = 0x4, `L4_IRQ_F_BOTH` = 0x8, `L4_IRQ_F_LEVEL_HIGH` = 0x3, `L4_IRQ_F_LEVEL_LOW`  
= 0x7,  
`L4_IRQ_F_POS_EDGE` = 0x1, `L4_IRQ_F_NEG_EDGE` = 0x5, `L4_IRQ_F_BOTH_EDGE` = 0x9, `L4_IRQ_F_MASK` = 0xf,  
`L4_IRQ_F_SET_WAKEUP` = 0x10, `L4_IRQ_F_CLEAR_WAKEUP` = 0x20 }  
*Interrupt attributes.*
- enum `L4_icu_opcode` {  
`L4_ICU_OP_BIND`, `L4_ICU_OP_UNBIND`, `L4_ICU_OP_INFO`, `L4_ICU_OP_MSI_INFO`,  
`L4_ICU_OP_UNMASK`, `L4_ICU_OP_MASK`, `L4_ICU_OP_SET_MODE` }  
*Opcodes to the ICU interface.*

## Functions

- `l4_msgtag_t l4_icu_bind (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW`  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4_msgtag_t l4_icu_bind_u (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4_msgtag_t l4_icu_unbind (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW`  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4_msgtag_t l4_icu_unbind_u (l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4_msgtag_t l4_icu_set_mode (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW`  
*Set interrupt mode.*
- `l4_msgtag_t l4_icu_set_mode_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode, l4_utcb_t *utcb) L4_NOTHROW`  
*Set interrupt mode.*
- `l4_msgtag_t l4_icu_info (l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW`  
*Get information about the capabilities of the ICU.*
- `l4_msgtag_t l4_icu_info_u (l4_cap_idx_t icu, l4_icu_info_t *info, l4_utcb_t *utcb) L4_NOTHROW`  
*Get information about the capabilities of the ICU.*
- `l4_msgtag_t l4_icu_msi_info (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info) L4_NOTHROW`  
*Get MSI info about IRQ.*
- `l4_msgtag_t l4_icu_msi_info_u (l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW`  
*Get MSI info about IRQ.*
- `l4_msgtag_t l4_icu_unmask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW`  
*Unmask an IRQ line.*
- `l4_msgtag_t l4_icu_unmask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW`  
*Acknowledge the given interrupt line.*
- `l4_msgtag_t l4_icu_mask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW`  
*Mask an IRQ line.*
- `l4_msgtag_t l4_icu_mask_u (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW`  
*Mask an IRQ line.*

### 15.309.1 Detailed Description

Interrupt controller.

Definition in file [icu.h](#).

## 15.310 icu.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029
00050 enum L4_icu_flags
00051 {
00059 L4_ICU_FLAG_MSI = 0x80000000,
00060 };
00061
00062
00067 enum L4_irq_mode
00068 {
00070 L4_IRQ_F_NONE = 0,
00071 L4_IRQ_F_LEVEL = 0x2,
00072 L4_IRQ_F_EDGE = 0x0,
00073 L4_IRQ_F_POS = 0x0,
00074 L4_IRQ_F_NEG = 0x4,
00075 L4_IRQ_F_BOTH = 0x8,
00076 L4_IRQ_F_LEVEL_HIGH = 0x3,
00077 L4_IRQ_F_LEVEL_LOW = 0x7,
00078 L4_IRQ_F_POS_EDGE = 0x1,
00079 L4_IRQ_F_NEG_EDGE = 0x5,
00080 L4_IRQ_F_BOTH_EDGE = 0x9,
00081 L4_IRQ_F_MASK = 0xf,
00084 L4_IRQ_F_SET_WAKEUP = 0x10,
00085 L4_IRQ_F_CLEAR_WAKEUP = 0x20,
00086 };
00087
00088
00093 enum L4_icu_opcode
00094 {
00100 L4_ICU_OP_BIND = 0,
00101
00107 L4_ICU_OP_UNBIND = 1,
00108
00114 L4_ICU_OP_INFO = 2,
00115
00121 L4_ICU_OP_MSI_INFO = 3,
00122
00128 L4_ICU_OP_UNMASK = 4,
00129
00135 L4_ICU_OP_MASK = 5,
00136
00142 L4_ICU_OP_SET_MODE = 6,
00143 };
00144
00145 enum L4_icu_ctl_op
00146 {
00147 L4_ICU_CTL_UNMASK = 0,
00148 L4_ICU_CTL_MASK = 1

```

```

00149 };
00150
00151
00159 typedef struct l4_icu_info_t
00160 {
00166 unsigned features;
00167
00171 unsigned nr_irqs;
00172
00176 unsigned nr_msis;
00177 } l4_icu_info_t;
00178
00180 typedef struct l4_icu_msi_info_t
00181 {
00183 l4_uint64_t msi_addr;
00185 l4_uint32_t msi_data;
00186 } l4_icu_msi_info_t;
00187
00204 L4_INLINE l4_msgtag_t
00205 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum,
00206 l4_cap_idx_t irq) L4_NOTHROW;
00207
00213 L4_INLINE l4_msgtag_t
00214 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum,
00215 l4_cap_idx_t irq,
00216 l4_utcb_t *utcb) L4_NOTHROW;
00217
00227 L4_INLINE l4_msgtag_t
00228 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum,
00229 l4_cap_idx_t irq) L4_NOTHROW;
00230
00236 L4_INLINE l4_msgtag_t
00237 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum,
00238 l4_cap_idx_t irq,
00239 l4_utcb_t *utcb) L4_NOTHROW;
00240
00250 L4_INLINE l4_msgtag_t
00251 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum,
00252 l4_umword_t mode) L4_NOTHROW;
00253
00259 L4_INLINE l4_msgtag_t
00260 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum,
00261 l4_umword_t mode,
00262 l4_utcb_t *utcb) L4_NOTHROW;
00263
00274 L4_INLINE l4_msgtag_t
00275 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info)
00276 L4_NOTHROW;
00277
00283 L4_INLINE l4_msgtag_t
00284 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00285 l4_utcb_t *utcb) L4_NOTHROW;
00286
00293 L4_INLINE l4_msgtag_t
00294 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum,
00295 l4_uint64_t source,
00296 l4_icu_msi_info_t *msi_info) L4_NOTHROW;
00297
00303 L4_INLINE l4_msgtag_t
00304 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum,
00305 l4_uint64_t source,
00306 l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb)
00307 L4_NOTHROW;
00308
00320 L4_INLINE l4_msgtag_t
00321 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum,
00322 l4_umword_t *label,
00323 l4_timeout_t to) L4_NOTHROW;
00324
00330 L4_INLINE l4_msgtag_t
00331 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum,
00332 l4_umword_t *label,
00333 l4_timeout_t to, l4_utcb_t *utcb)
00334 L4_NOTHROW;
00335
00345 L4_INLINE l4_msgtag_t
00346 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum,
00347 l4_umword_t *label,
00348 l4_timeout_t to) L4_NOTHROW;
00349
00355 L4_INLINE l4_msgtag_t
00356 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum,
00357 l4_umword_t *label,
00358 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00359
00362 L4_INLINE l4_msgtag_t

```

```

00363 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00364 l4_umword_t *label, l4_timeout_t to,
00365 l4_utcb_t *utcb) L4_NOTHROW;
00366
00367
00368 /*****
00369 * Implementations
00370 */
00371
00372 L4_INLINE l4_msgtag_t
00373 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum,
00374 l4_cap_idx_t irq,
00375 l4_utcb_t *utcb) L4_NOTHROW
00376 {
00377 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00378 m->mr[0] = L4_ICU_OP_BIND;
00379 m->mr[1] = irqnum;
00380 m->mr[2] = l4_map_obj_control(0, 0);
00381 m->mr[3] = l4_obj_fpage(irq, 0, L4_FPAGE_RWX).raw;
00382 return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0),
00383 L4_IPC_NEVER);
00384 }
00385
00386 L4_INLINE l4_msgtag_t
00387 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum,
00388 l4_cap_idx_t irq,
00389 l4_utcb_t *utcb) L4_NOTHROW
00390 {
00391 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00392 m->mr[0] = L4_ICU_OP_UNBIND;
00393 m->mr[1] = irqnum;
00394 m->mr[2] = l4_map_obj_control(0, 0);
00395 m->mr[3] = l4_obj_fpage(irq, 0, L4_FPAGE_RWX).raw;
00396 return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0),
00397 L4_IPC_NEVER);
00398 }
00399
00400 L4_INLINE l4_msgtag_t
00401 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00402 l4_utcb_t *utcb) L4_NOTHROW
00403 {
00404 l4_msgtag_t res;
00405 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00406 m->mr[0] = L4_ICU_OP_INFO;
00407 res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
00408 L4_IPC_NEVER);
00409 info->features = m->mr[0];
00410 info->nr_irqs = m->mr[1];
00411 info->nr_msis = m->mr[2];
00412 return res;
00413 }
00414
00415 L4_INLINE l4_msgtag_t
00416 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum,
00417 l4_uint64_t source,
00418 l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb)
00419 L4_NOTHROW
00420 {
00421 l4_msgtag_t res;
00422 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00423 m->mr[0] = L4_ICU_OP_MSI_INFO;
00424 m->mr[1] = irqnum;
00425 m->mr64[l4_utcb_mr64_idx(2)] = source;
00426 res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ,
00427 2 + 1 * sizeof(l4_uint64_t)
00428 / sizeof(l4_umword_t),
00429 0, 0), L4_IPC_NEVER);
00430 if (L4_UNLIKELY(l4_msgtag_has_error(res)))
00431 return res;
00432 if (L4_UNLIKELY(l4_msgtag_words(res) * sizeof(
00433 l4_umword_t) < sizeof(*msi_info)))
00434 return res;
00435 __builtin_memcpy(msi_info, &m->mr[0], sizeof(*msi_info));
00436 return res;
00437 }
00438
00439 L4_INLINE l4_msgtag_t
00440 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum,
00441 l4_umword_t mode,
00442 l4_utcb_t *utcb) L4_NOTHROW
00443 {
00444 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00445 m->mr[0] = L4_ICU_OP_SET_MODE;
00446 m->mr[1] = irqnum;
00447 m->mr[2] = mode;

```

```

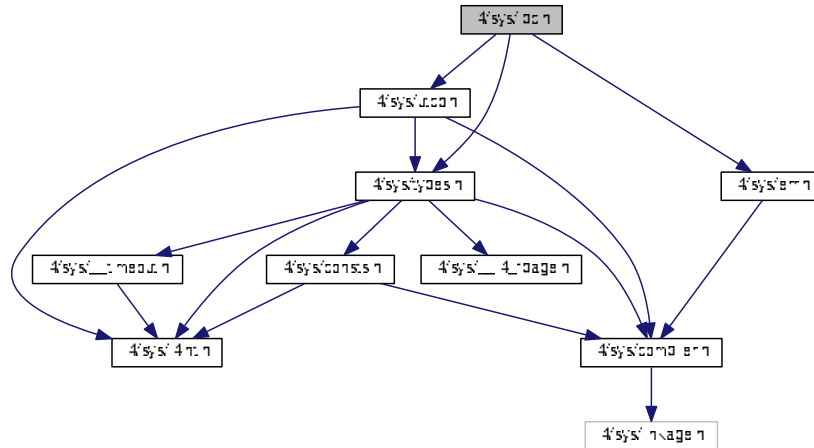
00441 return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 3, 0, 0),
00442 L4_IPC_NEVER);
00443 }
00444 L4_INLINE l4_msgtag_t
00445 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00446 l4_umword_t *label, l4_timeout_t to,
00447 l4_utcb_t *utcb) L4_NOTHROW
00448 {
00449 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00450 m->mr[0] = L4_ICU_OP_UNMASK + op;
00451 m->mr[1] = irqnum;
00452 if (label)
00453 return l4_ipc_send_and_wait(icu, utcb, l4_msgtag(
00454 L4_PROTO_IRQ, 2, 0, 0),
00455 label, to);
00456 else
00457 return l4_ipc_send(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0), to);
00458 }
00459 L4_INLINE l4_msgtag_t
00460 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum,
00461 l4_umword_t *label,
00462 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00463 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, utcb); }
00464 L4_INLINE l4_msgtag_t
00465 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum,
00466 l4_umword_t *label,
00467 l4_timeout_t to, l4_utcb_t *utcb)
00468 L4_NOTHROW
00469 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, utcb); }
00470
00471
00472 L4_INLINE l4_msgtag_t
00473 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum,
00474 l4_cap_idx_t irq) L4_NOTHROW
00475 { return l4_icu_bind_u(icu, irqnum, irq, l4_utcb()); }
00476 L4_INLINE l4_msgtag_t
00477 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum,
00478 l4_cap_idx_t irq) L4_NOTHROW
00479 { return l4_icu_unbind_u(icu, irqnum, irq, l4_utcb()); }
00480 L4_INLINE l4_msgtag_t
00481 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info)
00482 L4_NOTHROW
00483 { return l4_icu_info_u(icu, info, l4_utcb()); }
00484 L4_INLINE l4_msgtag_t
00485 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum,
00486 l4_uint64_t source,
00487 l4_icu_msi_info_t *msi_info) L4_NOTHROW
00488 { return l4_icu_msi_info_u(icu, irqnum, source, msi_info,
00489 l4_utcb()); }
00489 L4_INLINE l4_msgtag_t
00490 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum,
00491 l4_umword_t *label,
00492 l4_timeout_t to) L4_NOTHROW
00493 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, l4_utcb()); }
00494 L4_INLINE l4_msgtag_t
00495 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum,
00496 l4_umword_t *label,
00497 l4_timeout_t to) L4_NOTHROW
00498 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, l4_utcb()); }
00499 L4_INLINE l4_msgtag_t
00500 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum,
00501 l4_umword_t mode) L4_NOTHROW
00502 { return l4_icu_set_mode_u(icu, irqnum, mode, l4_utcb()); }
00503 }

```

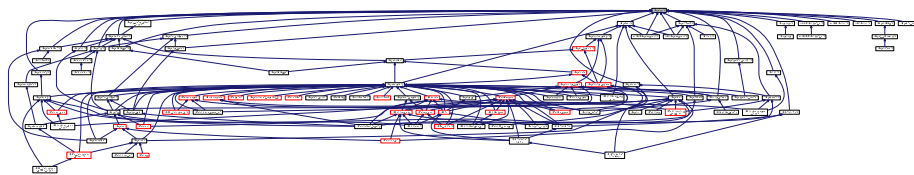
## 15.311 l4/sys/ipc.h File Reference

Common IPC interface.

```
#include <linux/sys/types.h>
#include <linux/sys/utcb.h>
#include <linux/sys/err.h>
Include dependency graph for ipc.h:
```



This graph shows which files directly or indirectly include this file:



## Enumerations

```
enum l4_ipc_tcr_error_t {
 L4_IPC_ERROR_MASK = 0x1F, L4_IPC_SND_ERR_MASK = 0x01, L4_IPC_ENOT_EXISTENT = 0x04,
 L4_IPC_RETIMEOUT = 0x03,
 L4_IPC_SETIMEOUT = 0x02, L4_IPC_RECANCELED = 0x07, L4_IPC_SECANCELED = 0x06, L4_IPC_←
 REMAPFAILED = 0x11,
 L4_IPC_SEMAPFAILED = 0x10, L4_IPC_RESNDPFTO = 0x0b, L4_IPC_SESNDPFTO = 0x0a, L4_IPC_←
 RERCVPFTO = 0x0d,
 L4_IPC_SERCVPFTO = 0x0c, L4_IPC_REABORTED = 0x0f, L4_IPC_SEABORTED = 0x0e, L4_IPC_RE←
 MSGCUT = 0x09,
 L4_IPC_SEMSGCUT = 0x08 }
}
```

*Error codes in the error TCR.*

## Functions

- `l4_umword_t l4_ipc_error (l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW`  
*Get the error code for an object invocation.*
- `long l4_error (l4_msgtag_t tag) L4_NOTHROW`

- Return error code of a system call return message tag.*

  - int `l4_ipc_is_snd_error` (`l4_utcb_t *utcb`) `L4_NOTHROW`  
*Returns whether an error occurred in send phase of an invocation.*
  - int `l4_ipc_is_rcv_error` (`l4_utcb_t *utcb`) `L4_NOTHROW`  
*Returns whether an error occurred in receive phase of an invocation.*
  - int `l4_ipc_error_code` (`l4_utcb_t *utcb`) `L4_NOTHROW`  
*Get the error condition of the last invocation from the TCR.*
  - long `l4_ipc_to_errno` (unsigned long `ipc_error_code`) `L4_NOTHROW`  
*Get a negative error code for the given IPC error code.*
  - `l4_msgtag_t l4_ipc_send` (`l4_cap_idx_t` `dest`, `l4_utcb_t *utcb`, `l4_msgtag_t` `tag`, `l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Send a message to an object (do **not** wait for a reply).*
  - `l4_msgtag_t l4_ipc_wait` (`l4_utcb_t *utcb`, `l4_umword_t *label`, `l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Wait for an incoming message from any possible sender.*
  - `l4_msgtag_t l4_ipc_receive` (`l4_cap_idx_t` `object`, `l4_utcb_t *utcb`, `l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Wait for a message from a specific source.*
  - `l4_msgtag_t l4_ipc_call` (`l4_cap_idx_t` `object`, `l4_utcb_t *utcb`, `l4_msgtag_t` `tag`, `l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Object call (usual invocation).*
  - `l4_msgtag_t l4_ipc_reply_and_wait` (`l4_utcb_t *utcb`, `l4_msgtag_t` `tag`, `l4_umword_t *label`, `l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Reply and wait operation (uses the reply capability).*
  - `l4_msgtag_t l4_ipc_send_and_wait` (`l4_cap_idx_t` `dest`, `l4_utcb_t *utcb`, `l4_msgtag_t` `tag`, `l4_umword_t *label`, `l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Send a message and do an open wait.*
  - `L4_ALWAYS_INLINE l4_ipc` (`l4_cap_idx_t` `dest`, `l4_utcb_t *utcb`, `l4_umword_t` `flags`, `l4_umword_t` `label`, `l4_msgtag_t` `tag`, `l4_umword_t *rlabel`, `l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Generic L4 object invocation.*
  - `l4_msgtag_t l4_ipc_sleep` (`l4_timeout_t` `timeout`) `L4_NOTHROW`  
*Sleep for an amount of time.*
  - int `l4_sndfpage_add` (`l4_fpage_t` `const snd_fpage`, unsigned long `snd_base`, `l4_msgtag_t *tag`) `L4_NOTHROW`  
*Add a flex-page to be sent to the UTCB.*

### 15.311.1 Detailed Description

Common IPC interface.

Definition in file `ipc.h`.

### 15.311.2 Function Documentation

#### 15.311.2.1 `l4_ipc_to_errno()`

```
long l4_ipc_to_errno (
 unsigned long ipc_error_code) [inline]
```

Get a negative error code for the given IPC error code.

## Parameters

|                             |                                                                                                               |
|-----------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>ipc_error_code</code> | IPC error code as delivered by the kernel. (or returned by the <a href="#">l4_ipc_error_code()</a> function). |
|-----------------------------|---------------------------------------------------------------------------------------------------------------|

## Returns

negative error code in the range of L4\_EIPC\_LO to L4\_EIPC\_HI.

Definition at line 441 of file [ipc.h](#).

References [L4\\_EIPC\\_LO](#).

## 15.312 ipc.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00027 #define __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/utcb.h>
00031 #include <l4/sys/err.h>
00032
00056 /*****
00057 *** IPC result checking
00058 *****/
00059
00075 enum l4_ipc_tcr_error_t
00076 {
00077 L4_IPC_ERROR_MASK = 0x1F,
00078 L4_IPC_SND_ERR_MASK = 0x01,
00080 L4_IPC_ENOT_EXISTENT = 0x04,
00083 L4_IPC_RETIMEOUT = 0x03,
00086 L4_IPC_SETIMEOUT = 0x02,
00089 L4_IPC_RECANCELED = 0x07,
00092 L4_IPC_SECANCELED = 0x06,
00095 L4_IPC_REMAPFAILED = 0x11,
00099 L4_IPC_SEMAPFAILED = 0x10,
00102 L4_IPC_RESNDPFTO = 0x0b,
00106 L4_IPC_SESNDPFTO = 0x0a,
00110 L4_IPC_RERCVPFTO = 0x0d,
00114 L4_IPC_SERCVPFTO = 0x0c,
00118 L4_IPC_REABORTED = 0x0f,
00121 L4_IPC_SEABORTED = 0x0e,
00124 L4_IPC_REMSGCUT = 0x09,
00128 L4_IPC_SEMSGCUT = 0x08,
00132 };
00133
00134
00145 L4_INLINE l4_umword_t
00146 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb)
00147 L4_NOTHROW;
00148
00155 L4_INLINE long

```



```

00156 l4_error(l4_msgtag_t tag) L4_NOTHROW;
00157
00158 L4_INLINE long
00159 l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00160
00161 /*****
00162 *** IPC results
00163 *****/
00164
00174 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb)
00175 L4_NOTHROW;
00176
00185 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb)
00186 L4_NOTHROW;
00187
00196 L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb)
00197 L4_NOTHROW;
00204 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code)
00205 L4_NOTHROW;
00206
00207 /*****
00208 *** IPC calls
00209 *****/
00210
00228 L4_INLINE l4_msgtag_t
00229 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb,
00230 l4_msgtag_t tag,
00231 l4_timeout_t timeout) L4_NOTHROW;
00232
00253 L4_INLINE l4_msgtag_t
00254 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00255 l4_timeout_t timeout) L4_NOTHROW;
00256
00277 L4_INLINE l4_msgtag_t
00278 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00279 l4_timeout_t timeout) L4_NOTHROW;
00280
00297 L4_INLINE l4_msgtag_t
00298 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb,
00299 l4_msgtag_t tag,
00300 l4_timeout_t timeout) L4_NOTHROW;
00301
00321 L4_INLINE l4_msgtag_t
00322 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00323 l4_umword_t *label, l4_timeout_t timeout)
00324 L4_NOTHROW;
00344 L4_INLINE l4_msgtag_t
00345 l4_ipc_send_and_wait(l4_cap_idx_t dest,
00346 l4_utcb_t *utcb, l4_msgtag_t tag,
00347 l4_umword_t *label, l4_timeout_t timeout)
00348 L4_NOTHROW;
00354 #if 0
00355
00365 L4_INLINE l4_msgtag_t
00366 l4_ipc_wait_next_period(l4_utcb_t *utcb,
00367 l4_umword_t *label,
00368 l4_timeout_t timeout);
00369
00370 #endif
00371
00386 L4_ALWAYS_INLINE l4_msgtag_t
00387 l4_ipc(l4_cap_idx_t dest,
00388 l4_utcb_t *utcb,
00389 l4_umword_t flags,
00390 l4_umword_t slabel,
00391 l4_msgtag_t tag,
00392 l4_umword_t *rlabel,
00393 l4_timeout_t timeout) L4_NOTHROW;
00394
00409 L4_INLINE l4_msgtag_t
00410 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;
00411
00424 L4_INLINE int
00425 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00426 l4_msgtag_t *tag) L4_NOTHROW;
00427
00428 /*
00429 * \internal
00430 * \ingroup l4_ipc_api
00431 */

```

```

00432 L4_INLINE int
00433 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00434 l4_msgtag_t *tag, l4_utcb_t *utcb)
00435 L4_NOTHROW;
00436
00437 /*****
00438 * Implementations
00439 *****/
00440
00441 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code)
00442 L4_NOTHROW
00443 { return -(L4_EIPC_LO + ipc_error_code); }
00444
00445 L4_INLINE l4_msgtag_t
00446 l4_ipc_call(l4_cap_idx_t dest, l4_utcb_t *utcb,
00447 l4_msgtag_t tag,
00448 l4_timeout_t timeout) L4_NOTHROW
00449 {
00450 return l4_ipc(dest, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
00451 }
00452
00453 L4_INLINE l4_msgtag_t
00454 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00455 l4_umword_t *label,
00456 l4_timeout_t timeout) L4_NOTHROW
00457 {
00458 return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0,
00459 tag, label, timeout);
00460 }
00461
00462 L4_INLINE l4_msgtag_t
00463 l4_ipc_send_and_wait(l4_cap_idx_t dest,
00464 l4_utcb_t *utcb,
00465 l4_msgtag_t tag,
00466 l4_umword_t *src,
00467 l4_timeout_t timeout) L4_NOTHROW
00468 {
00469 return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, src, timeout);
00470 }
00471
00472 L4_INLINE l4_msgtag_t
00473 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb,
00474 l4_msgtag_t tag,
00475 l4_timeout_t timeout) L4_NOTHROW
00476 {
00477 return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
00478 }
00479
00480 L4_INLINE l4_msgtag_t
00481 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *src,
00482 l4_timeout_t timeout) L4_NOTHROW
00483 {
00484 l4_msgtag_t t;
00485 t.raw = 0;
00486 return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, src, timeout);
00487 }
00488
00489 L4_INLINE l4_msgtag_t
00490 l4_ipc_receive(l4_cap_idx_t src, l4_utcb_t *utcb,
00491 l4_timeout_t timeout) L4_NOTHROW
00492 {
00493 l4_msgtag_t t;
00494 t.raw = 0;
00495 return l4_ipc(src, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
00496 }
00497
00498 L4_INLINE l4_msgtag_t
00499 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
00500 { return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
00501
00502 L4_INLINE l4_umword_t
00503 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb)
00504 L4_NOTHROW
00505 {
00506 if (!l4_msgtag_has_error(tag))
00507 return 0;
00508 return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
00509 }
00510
00511 L4_INLINE long
00512 l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
00513 {
00514 if (l4_msgtag_has_error(tag))
00515 return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error &
00516 L4_IPC_ERROR_MASK);
00517 }

```

```

00513 return l4_msgtag_label(tag);
00514 }
00515
00516 L4_INLINE long
00517 l4_error(l4_msgtag_t tag) L4_NOTHROW
00518 {
00519 return l4_error_u(tag, l4_utcb());
00520 }
00521
00522
00523 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u)
00524 L4_NOTHROW
00525 { return (l4_utcb_tcr_u(u)->error & 1) != 0; }
00526
00527 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u)
00528 L4_NOTHROW
00529 { return l4_utcb_tcr_u(u)->error & 1; }
00530
00531 L4_INLINE int l4_ipc_error_code(l4_utcb_t *u)
00532 L4_NOTHROW
00533 { return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }
00534
00535 /*
00536 * \internal
00537 * \ingroup l4_ipc_api
00538 */
00539 L4_INLINE int
00540 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00541 l4_msgtag_t *tag, l4_utcb_t *utcb)
00542 L4_NOTHROW
00543 {
00544 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00545 int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);
00546 if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
00547 return -L4_ENOMEM;
00548 v->mr[i] = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
00549 v->mr[i + 1] = snd_fpage.raw;
00550 *tag = l4_msgtag(l4_msgtag_label(*tag),
00551 l4_msgtag_words(*tag),
00552 l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
00553 return 0;
00554 }
00555
00556 L4_INLINE int
00557 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00558 l4_msgtag_t *tag) L4_NOTHROW
00559 {
00560 return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
00561 }
00562
00563 #endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */

```

## 15.313 arm/l4f/l4/sys/ipc.h File Reference

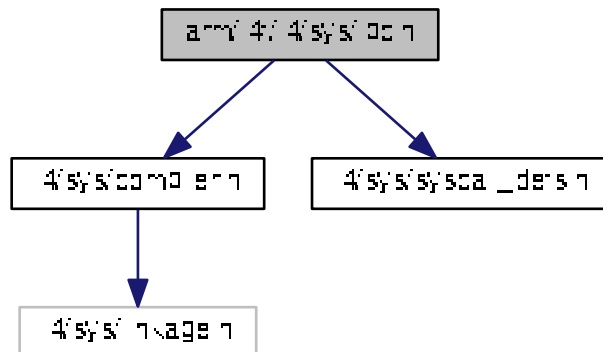
[L4](#) IPC System Calls, ARM.

```

#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>

```

Include dependency graph for ipc.h:



## Functions

- [l4\\_msgtag\\_t l4\\_ipc](#) ([l4\\_cap\\_idx\\_t](#) dest, [l4\\_utcb\\_t](#) \*utcb, [l4\\_umword\\_t](#) flags, [l4\\_umword\\_t](#) slabel, [l4\\_msgtag\\_t](#) \*tag, [l4\\_umword\\_t](#) \*rlabel, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
Generic [L4](#) object invocation.

### 15.313.1 Detailed Description

[L4](#) IPC System Calls, ARM.

Definition in file [ipc.h](#).

## 15.314 ipc.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include_next <l4/sys/ipc.h>
00027
00028 #ifdef __GNUC__
00029

```

```

00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/syscall_defs.h>
00032
00033 L4_INLINE l4_msgtag_t
00034 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00035 l4_umword_t flags,
00036 l4_umword_t slabel,
00037 l4_msgtag_t tag,
00038 l4_umword_t *rlabel,
00039 l4_timeout_t timeout) L4_NOTHROW
00040 {
00041 register l4_umword_t _dest __asm__("r2") = dest | flags;
00042 register l4_umword_t _timeout __asm__("r3") = timeout.raw;
00043 register l4_mword_t _tag __asm__("r0") = tag.raw;
00044 register l4_umword_t _label __asm__("r4") = slabel;
00045 (void)utcb;
00046
00047 __asm__ __volatile__
00048 ("mov lr, pc\n"
00049 "mov pc, %[sc]\n"
00050 :
00051 "+r" (_dest),
00052 "+r" (_timeout),
00053 "+r" (_label),
00054 "+r" (_tag)
00055 :
00056 [sc] "i" (L4_SYSCALL_INVOKE)
00057 :
00058 "cc", "memory", "lr");
00059
00060 if (rlabel)
00061 *rlabel = _label;
00062 tag.raw = _tag;
00063
00064 return tag;
00065 }
00066
00067 #endif //__GNUC__

```

## 15.315 x86/l4/l4/sys/ipc.h File Reference

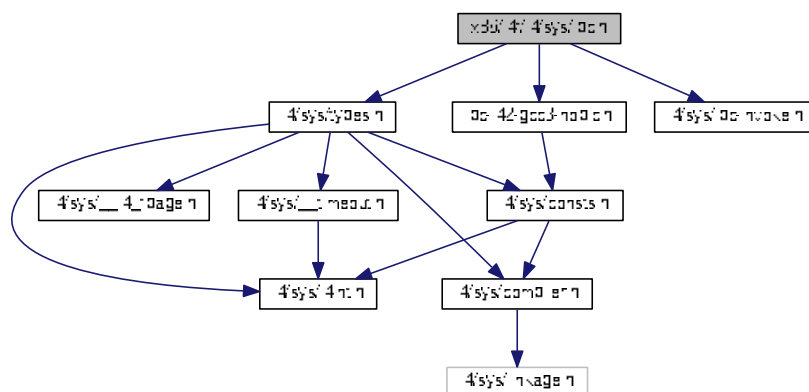
[L4](#) IPC System Calls, x86.

```

#include <l4/sys/types.h>
#include <l4/sys/ipc-invoke.h>
#include "ipc-l42-gcc3-nopic.h"

```

Include dependency graph for ipc.h:



### 15.315.1 Detailed Description

[L4 IPC System Calls, x86.](#)

Definition in file [ipc.h](#).

## 15.316 ipc.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Lars Reuther <reuther@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_IPC_H__
00026 #define __L4_IPC_H__
00027
00028 #include <l4/sys/types.h>
00029
00030 #include_next <l4/sys/ipc.h>
00031
00032 /***** Implementation *****/
00033
00034 #include <l4/sys/ipc-invoke.h>
00037 #include "ipc-l42-gcc3-nopic.h"
00038
00039 #endif /* !__L4_IPC_H__ */

```

## 15.317 l4/sys/ipc\_gate File Reference

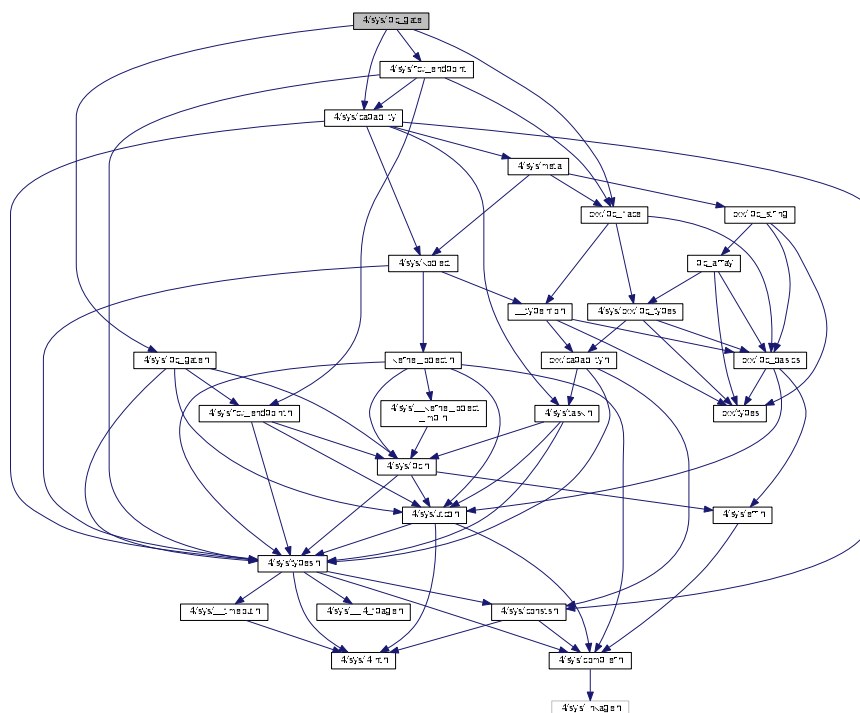
The C++ IPC gate interface.

```

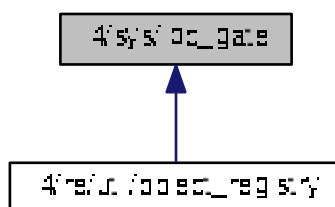
#include <l4/sys/ipc_gate.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for ipc\_gate:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::ipc\\_gate](#)  
The C++ IPC gate interface.

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 15.317.1 Detailed Description

The C++ IPC gate interface.

Definition in file [ipc\\_gate](#).

## 15.318 ipc\_gate

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003 * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021
00022 #include <l4/sys/ipc_gate.h>
00023 #include <l4/sys/capability>
00024 #include <l4/sys/rcv_endpoint>
00025 #include <l4/sys/cxx/ipc_iface>
00026
00027 namespace L4 {
00028
00029 class Thread;
00030
00031 class L4_EXPORT Ipc_gate :
00032 public Kobject_t<Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT,
00033 Type_info::Demand_t<1> >
00034 {
00035 public:
00036 L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00037 l4_msgtag_t, get_infos, (l4_umword_t *label));
00038 typedef L4::Typeid::Rpcs_sys<bind_thread_t, get_infos_t>
00039 Rpcs;
00040 };
00041
00042 }
```

## 15.319 l4/sys/ipc\_gate.h File Reference

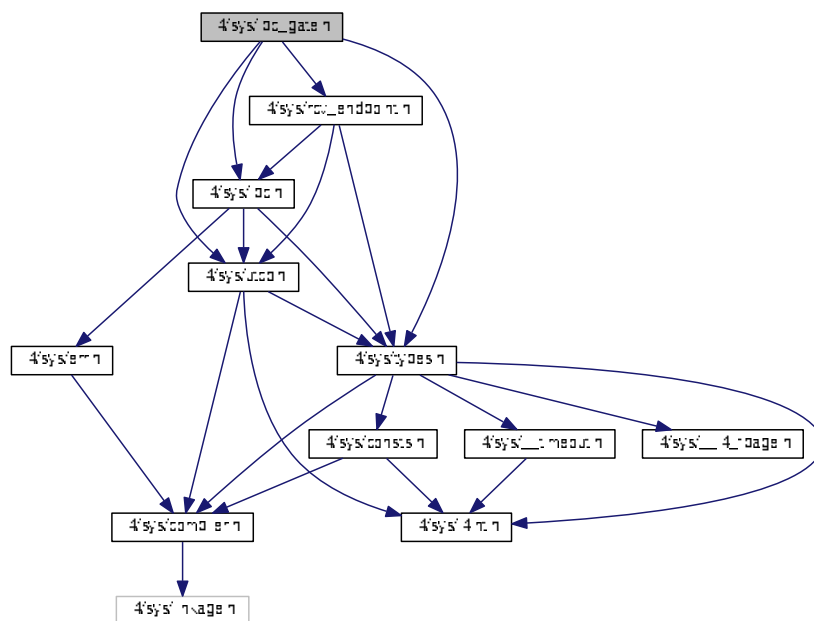
The C IPC gate interface.

```

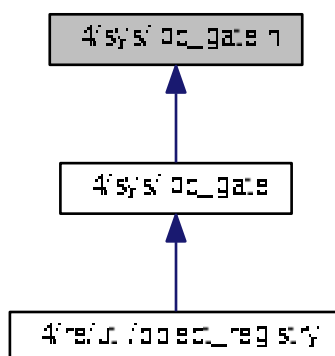
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/rcv_endpoint.h>
```



Include dependency graph for ipc\_gate.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_ipc_gate_ops` { `L4_IPC_GATE_BIND_OP` = 0x10, `L4_IPC_GATE_GET_INFO_OP` = 0x11 }
- Operations on the IPC-gate.*

## Functions

- `l4_msgtag_t l4_ipc_gate_bind_thread(l4_cap_idx_t gate, l4_cap_idx_t thread, l4_umword_t label)`  
Bind the IPC gate to a thread.
- `l4_msgtag_t l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label)`  
Get information about the IPC-gate.

### 15.319.1 Detailed Description

The C IPC gate interface.

IPC gates are used to create secure communication channels between threads. An IPC gate object can be created using the [Factory](#) interface. With `l4_ipc_gate_bind_thread()` a thread is bound to an IPC gate which then receives all messages sent to that IPC gate.

The `l4_ipc_gate_bind_thread()` call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The lower two bits of the *label* can be used to encode rights bits. The kernel combines these bits with the capability rights, so a programmer usually should not pick the lower two bits for the *label*. The *label* is only visible in the task which is running the thread the IPC gate was bound to and cannot be altered by the sender.

Definition in file [ipc\\_gate.h](#).

### 15.320 ipc\_gate.h

```

00001
00018 /*
00019 * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00020 * Alexander Warg <warg@os.inf.tu-dresden.de>
00021 * economic rights: Technische Universität Dresden (Germany)
00022 *
00023 * This file is part of TUD:OS and distributed under the terms of the
00024 * GNU General Public License 2.
00025 * Please see the COPYING-GPL-2 file for details.
00026 *
00027 * As a special exception, you may use this file as part of a free software
00028 * library without restriction. Specifically, if other files instantiate
00029 * templates or use macros or inline functions from this file, or you compile
00030 * this file and link it with other files to produce an executable, this
00031 * file does not by itself cause the resulting executable to be covered by
00032 * the GNU General Public License. This exception does not however
00033 * invalidate any other reasons why the executable file might be covered by
00034 * the GNU General Public License.
00035 */
00036 #pragma once
00037
00038 #include <l4/sys/utcb.h>
00039 #include <l4/sys/types.h>
00040 #include <l4/sys/rcv_endpoint.h>
00041
00058 L4_INLINE l4_msgtag_t
00059 l4_ipc_gate_bind_thread(l4_cap_idx_t gate,
00060 l4_cap_idx_t thread,
00061 l4_umword_t label)
00062 {
00063 L4_DEPRECATED("Use l4_rcv_ep_bind_thread().");
00064 }
00067 L4_INLINE l4_msgtag_t
00068 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate, l4_cap_idx_t thread,
00069 l4_umword_t label, l4_utcb_t *utcb)
00070 {
00071 L4_DEPRECATED("Use l4_rcv_ep_bind_thread_u().");
00072 }
00078 L4_INLINE l4_msgtag_t
00079 l4_ipc_gate_get_infos(l4_cap_idx_t gate,
00080 l4_umword_t *label);
00081
00085 L4_INLINE l4_msgtag_t
00086 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label,
00087 l4_utcb_t *utcb);

```

```

00087
00094 enum L4_ipc_gate_ops
00095 {
00096 L4_IPC_GATE_BIND_OP = 0x10,
00097 L4_IPC_GATE_GET_INFO_OP = 0x11,
00098 };
00099
00100
00101 /* IMPLEMENTATION -----*/
00102
00103 #include <l4/sys/ipc.h>
00104
00105 L4_INLINE l4_msgtag_t
00106 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
00107 l4_cap_idx_t thread, l4_umword_t label,
00108 l4_utcb_t *utcb)
00109 {
00110 return l4_rcv_ep_bind_thread_u(gate, thread, label, utcb);
00111 }
00112
00113 L4_INLINE l4_msgtag_t
00114 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label,
00115 l4_utcb_t *utcb)
00116 {
00117 l4_msgtag_t tag;
00118 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00119 m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
00120 tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
00121 L4_IPC_NEVER);
00122 if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
00123 *label = m->mr[0];
00124 return tag;
00125 }
00126
00127
00128
00129 L4_INLINE l4_msgtag_t
00130 l4_ipc_gate_bind_thread(l4_cap_idx_t gate,
00131 l4_cap_idx_t thread,
00132 l4_umword_t label)
00133 {
00134 return l4_rcv_ep_bind_thread_u(gate, thread, label, l4_utcb());
00135 }
00136
00137 L4_INLINE l4_msgtag_t
00138 l4_ipc_gate_get_infos(l4_cap_idx_t gate,
00139 l4_umword_t *label)
00140 {
00141 return l4_ipc_gate_get_infos_u(gate, label, l4_utcb());
00142 }

```

## 15.321 l4/sys/irq File Reference

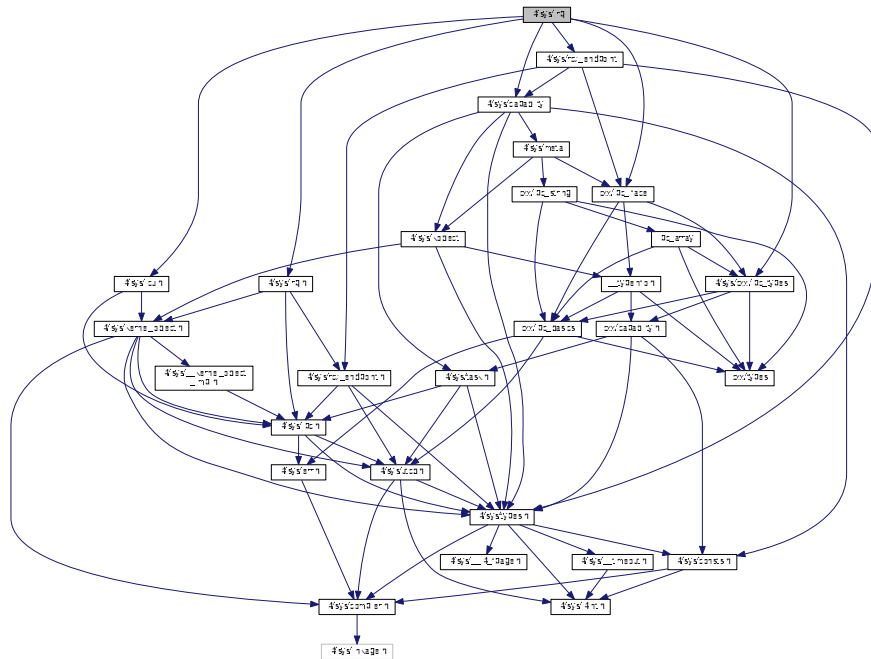
C++ Irq interface.

```

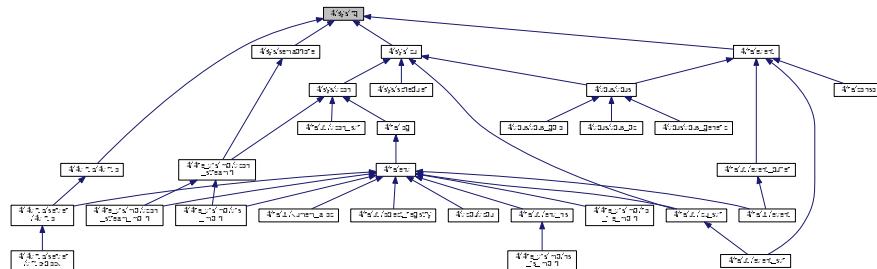
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_types>

```

Include dependency graph for `irq`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Irq_eoi`  
*Interface for sending an acknowledge message to an object.*
- struct `L4::Triggerable`  
*Interface that allows an object to be triggered by some source.*
- class `L4::Irq`  
*C++ `Irq` interface.*
- struct `L4::Irq_mux`  
*IRQ multiplexer for shared IRQs.*
- class `L4::Icu`  
*C++ `Icu` interface.*
- class `L4::Icu::Info`  
*This class encapsulates information about an ICU.*

## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

### 15.321.1 Detailed Description

C++ Irq interface.

Definition in file [irq](#).

## 15.322 irq

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/icu.h>
00028 #include <l4/sys/irq.h>
00029 #include <l4/sys/capability>
00030 #include <l4/sys/rcv_endpoint>
00031 #include <l4/sys/cxx/ipc_iface>
00032 #include <l4/sys/cxx/ipc_types>
00033
00034 namespace L4 {
00035
00043 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00044 {
00045 public:
00065 l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
00066 l4_timeout_t to = L4_IPC_NEVER,
00067 l4_utcb_t *utcb = l4_utcb()) throw()
00068 {
00069 return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00070 }
00071 };
00072
00078 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00079 {
00093 l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) throw()
00094 { return l4_irq_trigger_u(cap(), utcb); }
00095 };
00096
00117 class Irq : public Kobject_2t<Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER>
00118 {
00119 public:
00120 using Triggerable::unmask;
00121
00138 l4_msgtag_t attach(l4_umword_t label,
00139 Cap<Thread> const &thread = Cap<Thread>::Invalid,
00140 l4_utcb_t *utcb = l4_utcb()) throw()
00141 {
00142 L4_DEPRECATED("Use bind_thread(thread, label).")
00143 {
00144 #pragma GCC diagnostic push
00145 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00146 return l4_irq_attach_u(cap(), label, thread.cap(), utcb);
00147 }
00148 }
00149 };

```

```

00146 #pragma GCC diagnostic pop
00147 }
00148
00156 l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) throw()
00157 { return l4_irq_detach_u(cap(), utcb); }
00158
00159
00171 l4_msgtag_t receive(l4_timeout_t timeout =
L4_IPC_NEVER,
00172 l4_utcb_t *utcb = l4_utcb()) throw()
00173 { return l4_irq_receive_u(cap(), timeout, utcb); }
00174
00184 l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout =
L4_IPC_NEVER,
00185 l4_utcb_t *utcb = l4_utcb()) throw()
00186 { return unmask(-1, label, timeout, utcb); }
00187
00207 l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) throw()
00208 { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00209 };
00210
00224 struct Irq_mux : Kobject_t<Irq_mux, Triggerable, L4_PROTO_IRQ_MUX>
00225 {
00239 l4_msgtag_t chain(Cap<Triggerable> const &slave,
00240 l4_utcb_t *utcb = l4_utcb()) throw()
00241 { return l4_irq_mux_chain_u(cap(), slave.cap(), utcb); }
00242 };
00243
00244
00262 class Icu :
00263 public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00264 Type_info::Demand_t<1> >
00265 {
00266 public:
00267 enum Mode
00268 {
00269 F_none = L4_IRQ_F_NONE,
00270 F_level_high = L4_IRQ_F_LEVEL_HIGH,
00271 F_level_low = L4_IRQ_F_LEVEL_LOW,
00272 F_pos_edge = L4_IRQ_F_POS_EDGE,
00273 F_neg_edge = L4_IRQ_F_NEG_EDGE,
00274 F_both_edge = L4_IRQ_F_BOTH_EDGE,
00275 F_mask = L4_IRQ_F_MASK,
00276
00277 F_set_wakeup = L4_IRQ_F_SET_WAKEUP,
00278 F_clear_wakeup = L4_IRQ_F_CLEAR_WAKEUP,
00279 };
00280
00281 enum Flags
00282 {
00283 F_msi = L4_ICU_FLAG_MSI
00284 };
00285
00289 class Info : public l4_icu_info_t
00290 {
00291 public:
00292 bool supports_msi() const { return features & F_msi; }
00293 };
00294
00310 l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00311 l4_utcb_t *utcb = l4_utcb()) throw()
00312 { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00313
00314 L4_RPC_NF_OP(L4_ICU_OP_BIND,
00315 l4_msgtag_t, bind, (l4_umword_t irqnum,
Ipc::Cap<Irq> irq));
00316
00326 l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00327 l4_utcb_t *utcb = l4_utcb()) throw()
00328 { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00329
00330 L4_RPC_NF_OP(L4_ICU_OP_UNBIND,
00331 l4_msgtag_t, unbind, (l4_umword_t irqnum,
Ipc::Cap<Irq> irq));
00332
00341 l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb =
l4_utcb()) throw()
00342 { return l4_icu_info_u(cap(), info, utcb); }
00343
00344 struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00345 L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00346
00359 L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00360 l4_msgtag_t, msi_info, (l4_umword_t irqnum,
l4_uint64_t source,
00361 l4_icu_msi_info_t *msi_info));
00362

```

```

00366 l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00367 l4_timeout_t to, l4_utcb_t *utcb =
00368 l4_utcb()) throw()
00369 { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }
00369
00384 l4_msgtag_t mask(unsigned irqnum,
00385 l4_umword_t *label = 0,
00386 l4_timeout_t to = L4_IPC_NEVER,
00387 l4_utcb_t *utcb = l4_utcb()) throw()
00388 { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00389
00390 L4_RPC_NF_OP(L4_ICU_OP_MASK, l4_msgtag_t, mask, (
00391 l4_umword_t irqnum),
00392 L4::Ipc::Send_only);
00392
00393
00394 L4_RPC_NF_OP(L4_ICU_OP_UNMASK, l4_msgtag_t,
00395 unmask, (l4_umword_t irqnum),
00396 L4::Ipc::Send_only);
00396
00406 l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00407 l4_utcb_t *utcb = l4_utcb()) throw()
00408 { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }
00409
00410 L4_RPC_NF_OP(L4_ICU_OP_SET_MODE,
00411 l4_msgtag_t, set_mode, (l4_umword_t irqnum,
00412 l4_umword_t mode));
00412
00413 typedef L4::Typeid::Rpcsys<
00414 bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00415 > Rpc;
00416 };
00417
00418 }

```

## 15.323 l4/sys/kernel\_object.h File Reference

Kernel object system calls.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__kernel_object_impl.h>
#include <l4/sys/ipc.h>

```





```

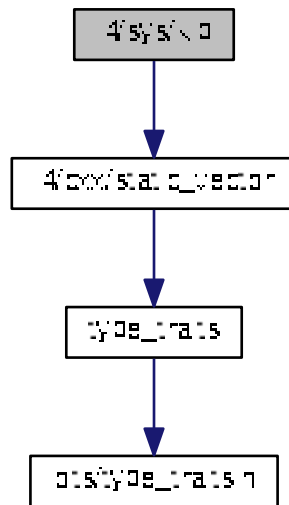
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4SYS__KERNEL_OBJECT_H__
00025 #define __L4SYS__KERNEL_OBJECT_H__
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/compiler.h>
00029 #include <l4/sys/utcb.h>
00030
00049 L4_INLINE l4_msgtag_t
00050 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag,
00051 l4_utcb_t *utcb) L4_NOTHROW;
00052
00053 /*****
00054 * Implementation
00055 *****/
00056
00057 #include <l4/sys/__kernel_object_impl.h>
00058 #include <l4/sys/ipc.h>
00059
00060 enum L4_kobject_op {
00061 L4_KOBJECT_OP_DEC_REFCNT = 0,
00062 L4_KOBJECT_OP_REGISTER_IRQ,
00063 };
00064
00065 L4_INLINE l4_msgtag_t
00066 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff,
00067 l4_utcb_t *u) L4_NOTHROW;
00068
00069 L4_INLINE l4_msgtag_t
00070 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff)
00071 L4_NOTHROW;
00072
00073 L4_INLINE l4_msgtag_t
00074 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff,
00075 l4_utcb_t *u) L4_NOTHROW
00076 {
00077 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00078 m->mr[0] = L4_KOBJECT_OP_DEC_REFCNT;
00079 m->mr[1] = diff;
00080 return l4_ipc_call(obj, u, l4_msgtag(L4_PROTO_KOBJECT, 2, 0, 0),
00081 L4_IPC_NEVER);
00082 }
00083
00084 L4_INLINE l4_msgtag_t
00085 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff)
00086 L4_NOTHROW
00087 {
00088 return l4_kobject_dec_refcnt_u(obj, diff, l4_utcb());
00089 }
00090
00091 #endif /* ! __L4SYS__KERNEL_OBJECT_H__ */

```

## 15.325 l4/sys/kip File Reference

```
#include <l4/cxx/static_vector>
```

Include dependency graph for kip:



## Data Structures

- class [L4::Kip::Mem\\_desc](#)

*Memory descriptors stored in the kernel interface page.*

## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

## 15.325.1 Detailed Description

L4::Kip class, memory descriptors.

### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [kip](#).

## 15.326 kip

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00010 /*
00011 * (c) 2008-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 #ifndef L4_SYS_KIP_H__
00028 #define L4_SYS_KIP_H__
00029
00030 #include <l4/cxx/static_vector>
00031
00032 /* C++ version of memory descriptors */
00033
00043 namespace L4
00044 {
00045 namespace Kip
00046 {
00053 class Mem_desc
00054 {
00055 public:
00059 enum Mem_type
00060 {
00061 Undefined = 0x0,
00062 Conventional = 0x1,
00063 Reserved = 0x2,
00064 Dedicated = 0x3,
00065 Shared = 0x4,
00066
00067 Info = 0xd,
00068 Bootloader = 0xe,
00069 Arch = 0xf
00070 };
00071
00075 enum Info_sub_type
00076 {
00077 Info_acpi_rsdp = 0
00078 };
00079
00080 private:
00081 unsigned long _l, _h;
00082
00083 static unsigned long &memory_info(void *kip) throw()
00084 { return *((unsigned long *)kip + 21); }
00085
00086 static unsigned long memory_info(void const *kip) throw()
00087 { return *((unsigned long const *)kip + 21); }
00088
00089 public:
00097 static Mem_desc *first(void *kip) throw()
00098 {
00099 return (Mem_desc *)((char *)kip
00100 + (memory_info(kip) >> ((sizeof(unsigned long) / 2) * 8)));
00101 }
00102
00103 static Mem_desc const *first(void const *kip) throw()
00104 {
00105 return (Mem_desc const *)((char const *)kip
00106 + (memory_info(kip) >> ((sizeof(unsigned long) / 2) * 8)));
00107 }
00108
00116 static unsigned long count(void const *kip) throw()
00117 {
00118 return memory_info(kip)
00119 & ((1UL << ((sizeof(unsigned long) / 2) * 8)) - 1);
00120 }
00121
00128 static void count(void *kip, unsigned count) throw()
00129 {
00130 unsigned long &mi = memory_info(kip);
00131 mi = (mi & ~((1UL << ((sizeof(unsigned long) / 2) * 8)) - 1)) | count;
00132 }
00133

```

```

00139 static inline cxx::static_vector<Mem_desc const>
00140 all(void const *kip)
00141 {
00142 return cxx::static_vector<Mem_desc const>(
00143 Mem_desc::first(kip),
00144 Mem_desc::count(kip));
00145 }
00146
00147 static inline cxx::static_vector<Mem_desc> all(void *kip)
00148 {
00149 return cxx::static_vector<Mem_desc>(
00150 Mem_desc::first(kip),
00151 Mem_desc::count(kip));
00152 }
00153
00154 Mem_desc(unsigned long start, unsigned long end,
00155 Mem_type t, unsigned char st = 0, bool virt = false) throw()
00156 : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00157 | (virt ? 0x0200 : 0x0)), _h(end | 0x3ffUL)
00158 {}
00159
00160 unsigned long start() const throw() { return _l & ~0x3ffUL; }
00161
00162 unsigned long end() const throw() { return _h | 0x3ffUL; }
00163
00164 unsigned long size() const throw() { return end() + 1 - start(); }
00165
00166 Mem_type type() const throw() { return (Mem_type)(_l & 0x0f); }
00167
00168 unsigned char sub_type() const throw() { return (_l >> 4) & 0x0f; }
00169
00170 unsigned is_virtual() const throw() { return _l & 0x200; }
00171
00172 void set(unsigned long start, unsigned long end,
00173 Mem_type t, unsigned char st = 0, bool virt = false) throw()
00174 {
00175 _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00176 | (virt?0x0200:0x0);
00177 _h = end | 0x3ffUL;
00178 }
00179 };
00180
00181 #endif

```

## 15.327 l4/sys/ktrace.h File Reference

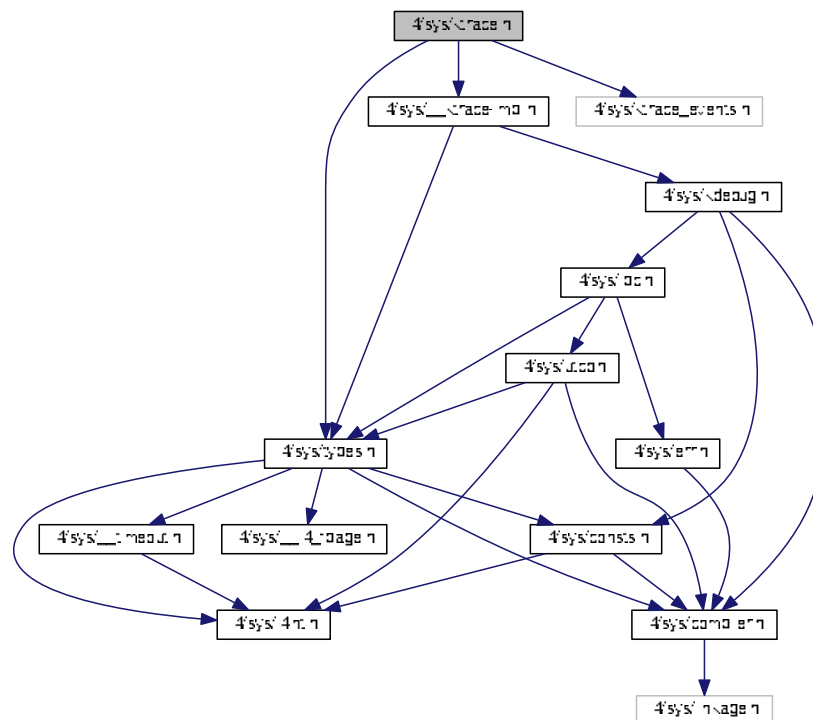
[L4](#) kernel event tracing.

```

#include <l4/sys/types.h>
#include <l4/sys/ktrace_events.h>
#include <l4/sys/__ktrace-impl.h>

```

Include dependency graph for ktrace.h:



## Data Structures

- struct `l4_tracebuffer_status_window_t`  
*Trace-buffer status window descriptor.*
- struct `l4_tracebuffer_status_t`  
*Trace-buffer status.*

## Enumerations

- enum {  
LOG\_EVENT\_CONTEXT\_SWITCH = 0, LOG\_EVENT\_IPC\_SHORTCUT = 1, LOG\_EVENT\_IRQ\_RAISED  
= 2, LOG\_EVENT\_TIMER\_IRQ = 3,  
LOG\_EVENT\_THREAD\_EX\_REGS = 4, LOG\_EVENT\_MAX\_EVENTS = 16 }  
*Log event types.*

## Functions

- `l4_tracebuffer_status_t * fiasco_tbuf_get_status` (void)  
*Return trace-buffer status.*
- `l4_addr_t fiasco_tbuf_get_status_phys` (void)  
*Return the physical address of the trace-buffer status struct.*
- `l4_umword_t fiasco_tbuf_log` (const char \*text)

Create new trace-buffer entry with describing <text>.

- `l4_umword_t fiasco_tbuf_log_3val` (const char \*text, `l4_umword_t` v1, `l4_umword_t` v2, `l4_umword_t` v3)

Create new trace-buffer entry with describing <text> and three additional values.

- `l4_umword_t fiasco_tbuf_log_binary` (const unsigned char \*data)

Create new trace-buffer entry with binary data.

- void `fiasco_tbuf_clear` (void)

Clear trace-buffer.

- void `fiasco_tbuf_dump` (void)

Dump trace-buffer to kernel console.

## 15.327.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [ktrace.h](#).

## 15.328 ktrace.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * 2015 Adam Lackorzynski <adam@l4re.org>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 /*****
00027 #ifndef __L4_KTRACE_H__
00028 #define __L4_KTRACE_H__
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/ktrace_events.h>
00032
00037 enum
00038 {
00039 LOG_EVENT_CONTEXT_SWITCH = 0,
00040 LOG_EVENT_IPC_SHORTCUT = 1,
00041 LOG_EVENT_IRQ_RAISED = 2,
00042 LOG_EVENT_TIMER_IRQ = 3,
00043 LOG_EVENT_THREAD_EX_REGS = 4,
00044 LOG_EVENT_MAX_EVENTS = 16,
00045 };
00046
00051 // keep in sync with fiasco/src/jabi/jdb_ktrace.cpp
00052 typedef struct
00053 {
00055 l4_tracebuffer_entry_t *tracebuffer;
00057 l4_umword_t size;
00059 volatile l4_uint64_t version;
00060 } l4_tracebuffer_status_window_t;
00061
00066 // keep in sync with fiasco/src/jabi/jdb_ktrace.cpp
00067 typedef struct
00068 {
00069 l4_tracebuffer_status_window_t window[2];
00071 volatile l4_tracebuffer_entry_t * current_entry;
00073 l4_uint32_t logevents[LOG_EVENT_MAX_EVENTS];

```

```

00074
00076 l4_uint32_t scaler_tsc_to_ns;
00078 l4_uint32_t scaler_tsc_to_us;
00080 l4_uint32_t scaler_ns_to_tsc;
00081
00083 volatile l4_uint32_t cnt_context_switch;
00085 volatile l4_uint32_t cnt_addr_space_switch;
00087 volatile l4_uint32_t cnt_shortcut_failed;
00089 volatile l4_uint32_t cnt_shortcut_success;
00091 volatile l4_uint32_t cnt_irq;
00093 volatile l4_uint32_t cnt_ipc_long;
00095 volatile l4_uint32_t cnt_page_fault;
00098 volatile l4_uint32_t cnt_io_fault;
00100 volatile l4_uint32_t cnt_task_create;
00102 volatile l4_uint32_t cnt_schedule;
00106 volatile l4_uint32_t cnt_iobmap_tlb_flush;
00107
00108 } l4_tracebuffer_status_t;
00109
00116 L4_INLINE l4_tracebuffer_status_t *
00117 fiasco_tbuf_get_status(void);
00118
00125 L4_INLINE l4_addr_t
00126 fiasco_tbuf_get_status_phys(void);
00127
00135 L4_INLINE l4_umword_t
00136 fiasco_tbuf_log(const char *text);
00137
00149 L4_INLINE l4_umword_t
00150 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1,
00151 l4_umword_t v2, l4_umword_t v3);
00151
00159 L4_INLINE l4_umword_t
00160 fiasco_tbuf_log_binary(const unsigned char *data);
00161
00166 L4_INLINE void
00167 fiasco_tbuf_clear(void);
00168
00173 L4_INLINE void
00174 fiasco_tbuf_dump(void);
00175
00176 #include <l4/sys/__ktrace-impl.h>
00177
00178 #endif

```

## 15.329 l4/sys/l4int.h File Reference

Fixed sized integer types, generic version.

This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef signed char [l4\\_int8\\_t](#)  
*Signed 8bit value.*
- typedef unsigned char [l4\\_uint8\\_t](#)  
*Unsigned 8bit value.*
- typedef signed short int [l4\\_int16\\_t](#)  
*Signed 16bit value.*
- typedef unsigned short int [l4\\_uint16\\_t](#)  
*Unsigned 16bit value.*
- typedef signed int [l4\\_int32\\_t](#)

- *Signed 32bit value.*
- typedef unsigned int [l4\\_uint32\\_t](#)
- *Unsigned 32bit value.*
- typedef signed long long [l4\\_int64\\_t](#)
- *Signed 64bit value.*
- typedef unsigned long long [l4\\_uint64\\_t](#)
- *Unsigned 64bit value.*
- typedef unsigned long [l4\\_addr\\_t](#)
- *Address type.*
- typedef signed long [l4\\_mword\\_t](#)
- *Signed machine word.*
- typedef unsigned long [l4\\_umword\\_t](#)
- *Unsigned machine word.*
- typedef [l4\\_uint64\\_t](#) [l4\\_cpu\\_time\\_t](#)
- *CPU clock type.*
- typedef [l4\\_uint64\\_t](#) [l4\\_kernel\\_clock\\_t](#)
- *Kernel clock type.*

### 15.329.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file [l4int.h](#).

## 15.330 l4int.h

```

00001
00013 /*
00014 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00015 * Alexander Warg <warg@os.inf.tu-dresden.de>
00016 * economic rights: Technische Universität Dresden (Germany)
00017 *
00018 * This file is part of TUD:OS and distributed under the terms of the
00019 * GNU General Public License 2.
00020 * Please see the COPYING-GPL-2 file for details.
00021 *
00022 * As a special exception, you may use this file as part of a free software
00023 * library without restriction. Specifically, if other files instantiate
00024 * templates or use macros or inline functions from this file, or you compile
00025 * this file and link it with other files to produce an executable, this
00026 * file does not by itself cause the resulting executable to be covered by
00027 * the GNU General Public License. This exception does not however
00028 * invalidate any other reasons why the executable file might be covered by
00029 * the GNU General Public License.
00030 */
00031 #ifndef __L4_SYS_L4INT_H__
00032 #define __L4_SYS_L4INT_H__
00033
00034 /* fixed sized data types */
00035 typedef signed char l4_int8_t;
00036 typedef unsigned char l4_uint8_t;
00037 typedef signed short int l4_int16_t;
00038 typedef unsigned short int l4_uint16_t;
00039 typedef signed int l4_int32_t;
00040 typedef unsigned int l4_uint32_t;
00041 typedef signed long long l4_int64_t;
00042 typedef unsigned long long l4_uint64_t;
00043
00044 /* some common data types */
00045 typedef unsigned long l4_addr_t;
00046 //do-we-need-this?//typedef unsigned long l4_offs_t; /**< Address offset type \ingroup
00047 l4_basic_types */
00048
00049 typedef signed long l4_mword_t;
00050 typedef unsigned long l4_umword_t;
00051 typedef l4_uint64_t l4_cpu_time_t;
00052
00053 typedef l4_uint64_t l4_kernel_clock_t;
00054
00055 #endif /* !__L4_SYS_L4INT_H__ */

```



## 15.331 arm/l4/sys/l4int.h File Reference

Fixed sized integer types, arm version.

### Macros

- `#define L4_MWORD_BITS 32`  
*Size of machine words in bits.*

### Typedefs

- `typedef unsigned int l4_size_t`  
*Unsigned size type.*
- `typedef signed int l4_ssize_t`  
*Signed size type.*

#### 15.331.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

## 15.332 l4int.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include_next <l4/sys/l4int.h>
00027
00032
00033 #define L4_MWORD_BITS 32
00035 typedef unsigned int l4_size_t;
00036 typedef signed int l4_ssize_t;
00038

```

## 15.333 amd64/l4/sys/l4int.h File Reference

Fixed sized integer types, amd64 version.

## Macros

- `#define L4_MWORD_BITS 64`  
*Size of machine words in bits.*

## Typedefs

- `typedef unsigned long l4_size_t`  
*Unsigned size type.*
- `typedef signed long l4_ssize_t`  
*Signed size type.*

### 15.333.1 Detailed Description

Fixed sized integer types, amd64 version.

Definition in file [l4int.h](#).

## 15.334 l4int.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include_next <l4/sys/l4int.h>
00028
00033
00034 #define L4_MWORD_BITS 64
00036 typedef unsigned long l4_size_t;
00037 typedef signed long l4_ssize_t;
00039
```

### 15.335 x86/l4/sys/l4int.h File Reference

Fixed sized integer types, x86 version.

## Macros

- `#define L4_MWORD_BITS 32`  
*Size of machine words in bits.*

## Typedefs

- typedef unsigned int [l4\\_size\\_t](#)

*Unsigned size type.*

- typedef signed int [l4\\_ssize\\_t](#)

*Signed size type.*

### 15.335.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file [l4int.h](#).

## 15.336 l4int.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include_next <l4/sys/l4int.h>
00027
00032
00033 #define L4_MWORD_BITS 32
00035 typedef unsigned int l4_size_t;
00036 typedef signed int l4_ssize_t;
00038

```

### 15.337 l4/sys/memdesc.h File Reference

Memory description functions.



## Functions

- `l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get pointer to memory descriptors from KIP.*
- `unsigned l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get number of memory descriptors in KIP.*
- `void l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW`  
*Populate a memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get start address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get end address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get type of the memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get sub-type of memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get virtual flag of the memory descriptor.*

### 15.337.1 Detailed Description

Memory description functions.

Definition in file [memdesc.h](#).

## 15.338 memdesc.h

```

00001
00006 /*
00007 * (c) 2007-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4SYS_MEMDESC_H__
00025 #define __L4SYS_MEMDESC_H__
00026
00027 #include <l4/sys/kip.h>
00028
00044 enum l4_mem_type_t
00045 {
00046 l4_mem_type_undefined = 0x0,
00047 l4_mem_type_conventional = 0x1,
00048 l4_mem_type_reserved = 0x2,
00049 l4_mem_type_dedicated = 0x3,
00050 l4_mem_type_shared = 0x4,
00051
00052 l4_mem_type_info = 0xd,
00053 l4_mem_type_bootloader = 0xe,
00054 l4_mem_type_archspecific = 0xf,

```

```

00055 };
00056
00061 enum l4_mem_info_sub_type_t
00062 {
00063 l4_mem_info_acpi_rsdp = 0
00064 };
00065
00066
00074 typedef struct l4_kernel_info_mem_desc_t
00075 {
00077 l4_umword_t l;
00079 l4_umword_t h;
00080 } l4_kernel_info_mem_desc_t;
00081
00082
00087 L4_INLINE
00088 l4_kernel_info_mem_desc_t *
00089 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip)
00090 L4_NOTHROW;
00091
00097 L4_INLINE
00098 unsigned
00099 l4_kernel_info_get_num_mem_descs(
00100 l4_kernel_info_t *kip) L4_NOTHROW;
00101
00102
00112 L4_INLINE
00113 void
00114 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *
00115 md,
00116 l4_addr_t start,
00117 l4_addr_t end,
00118 unsigned type,
00119 unsigned virt,
00120 unsigned sub_type) L4_NOTHROW;
00121
00122
00127 L4_INLINE
00128 l4_umword_t
00129 l4_kernel_info_get_mem_desc_start(
00130 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00131
00132
00137 L4_INLINE
00138 l4_umword_t
00139 l4_kernel_info_get_mem_desc_end(
00140 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00141
00142
00147 L4_INLINE
00148 l4_umword_t
00149 l4_kernel_info_get_mem_desc_type(
00150 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00151
00152
00160 L4_INLINE
00161 l4_umword_t
00162 l4_kernel_info_get_mem_desc_subtype(
00163 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00164
00165
00170 L4_INLINE
00171 l4_umword_t
00172 l4_kernel_info_get_mem_desc_is_virtual(
00173 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00174
00175
00176 /*****
00177 * Implementations
00178 *****/
00179
00178 L4_INLINE
00179 l4_kernel_info_mem_desc_t *
00180 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip)
00181 L4_NOTHROW
00182 {
00183 return (l4_kernel_info_mem_desc_t *)(((l4_addr_t)kip)
00184 + (kip->mem_info >> (sizeof(l4_umword_t) * 4)));
00185 }
00186
00187 L4_INLINE
00188 unsigned
00189 l4_kernel_info_get_num_mem_descs(
00190 l4_kernel_info_t *kip) L4_NOTHROW
00191 {
00192 return kip->mem_info & ((1UL << (sizeof(l4_umword_t)*4)) -1);
00193 }
00194
00195 L4_INLINE
00196 void
00197 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *
00198 md,
00199 l4_addr_t start,
00200 l4_addr_t end,

```

```

00198 unsigned type,
00199 unsigned virt,
00200 unsigned sub_type) L4_NOTHROW
00201 {
00202 md->l = (start & ~0x3ffUL) | (type & 0x0f) | ((sub_type << 4) & 0x0f0)
00203 | (virt ? 0x200 : 0x0);
00204 md->h = end;
00205 }
00206
00207
00208 L4_INLINE
00209 l4_umword_t
00210 l4_kernel_info_get_mem_desc_start(
00211 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00212 {
00213 return md->l & ~0x3ffUL;
00214 }
00215 L4_INLINE
00216 l4_umword_t
00217 l4_kernel_info_get_mem_desc_end(
00218 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00219 {
00220 return md->h | 0x3ffUL;
00221 }
00222 L4_INLINE
00223 l4_umword_t
00224 l4_kernel_info_get_mem_desc_type(
00225 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00226 {
00227 return md->l & 0xf;
00228 }
00229 L4_INLINE
00230 l4_umword_t
00231 l4_kernel_info_get_mem_desc_subtype(
00232 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00233 {
00234 return (md->l & 0xf0) >> 4;
00235 }
00236 L4_INLINE
00237 l4_umword_t
00238 l4_kernel_info_get_mem_desc_is_virtual(
00239 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00240 {
00241 return md->l & 0x200;
00242 }
00243 #endif /* ! __L4SYS__MEMDESC_H__ */

```

## 15.339 l4/sys/meta File Reference

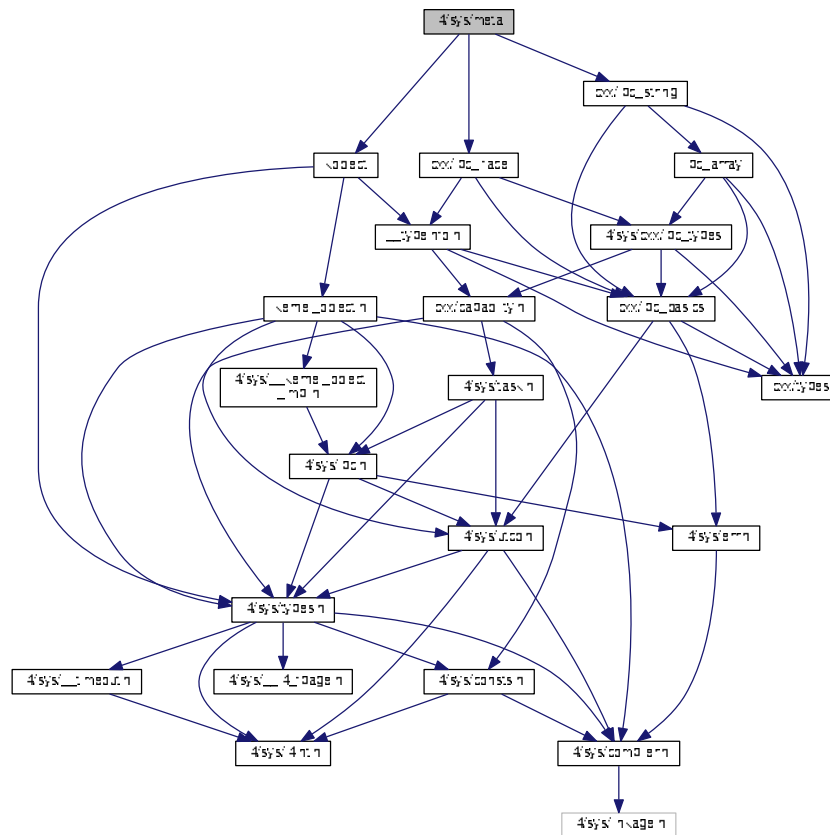
Meta interface for getting dynamic type information about objects behind capabilities.

```

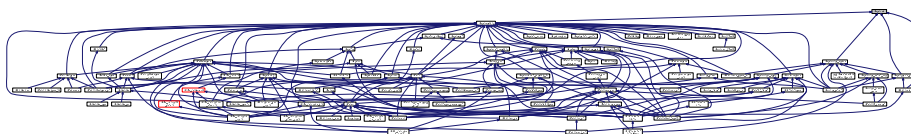
#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"

```

Include dependency graph for meta:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Meta](#)

*Meta* interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

## Namespaces

- [L4](#)

*L4* low-level kernel interface.



### 15.339.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file [meta](#).

## 15.340 meta

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004 * economic rights: Technische Universität Dresden (Germany)
00005 *
00006 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU General Public License 2.
00008 * Please see the COPYING-GPL-2 file for details.
00009 *
00010 * As a special exception, you may use this file as part of a free software
00011 * library without restriction. Specifically, if other files instantiate
00012 * templates or use macros or inline functions from this file, or you compile
00013 * this file and link it with other files to produce an executable, this
00014 * file does not by itself cause the resulting executable to be covered by
00015 * the GNU General Public License. This exception does not however
00016 * invalidate any other reasons why the executable file might be covered by
00017 * the GNU General Public License.
00018 */
00019 #pragma once
00020 #include "kobject"
00021 #include "cxx/ipc_iface"
00022 #include "cxx/ipc_string"
00023 namespace L4 {
00024 class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00025 {
00026 public:
00027 L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00028 L4_INLINE_RPC(l4_msgtag_t, interface, (
00029 l4_umword_t idx, long *proto,
00030 L4::Ipc::String<char> *name));
00031 L4_INLINE_RPC(l4_msgtag_t, supports, (
00032 l4_mword_t protocol));
00033 typedef L4::Typeid::Rpc<num_interfaces_t, interface_t, supports_t>
00034 Rpc;
00035 };
00036 }
```

## 15.341 l4/sys/pager File Reference

Pager and lo\_pager C++ interface.

```

#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
```



### 15.341.1 Detailed Description

Pager and `Io_pager` C++ interface.

Definition in file [pager](#).

## 15.342 pager

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019
00020 #include <l4/sys/capability>
00021 #include <l4/sys/types.h>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/cxx/ipc_iface>
00024
00025 namespace L4 {
00026
00027 class L4_EXPORT Io_pager :
00028 public Kobject_0t<Io_pager, L4_PROTO_IO_PAGE_FAULT>
00029 {
00030 public:
00031 L4_INLINE_RPC(
00032 l4_msgtag_t, io_page_fault, (l4_fpage_t io_pfa,
00033 l4_umword_t pc,
00034 L4::Ipc::Opt<l4_mword_t &> result,
00035 L4::Ipc::Rcv_fpage rwin,
00036 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp)
00037);
00038
00039 typedef L4::Typeid::Rpc_nocode<io_page_fault_t>
00040 Rpccs;
00041 };
00042
00043 class L4_EXPORT Pager :
00044 public Kobject_t<Pager, Io_pager, L4_PROTO_PAGE_FAULT>
00045 {
00046 public:
00047 L4_INLINE_RPC(
00048 l4_msgtag_t, page_fault, (l4_umword_t pfa,
00049 l4_umword_t pc,
00050 L4::Ipc::Opt<l4_mword_t &> result,
00051 L4::Ipc::Rcv_fpage rwin,
00052 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp))
00053);
00054
00055 typedef L4::Typeid::Rpc_nocode<page_fault_t>
00056 Rpccs;
00057 };
00058
00059 }
00060

```

## 15.343 l4/sys/platform\_control File Reference

Platform control object.



## 15.344 platform\_control

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008 * Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/sys/platform_control.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031
00032 namespace L4 {
00033
00046 class L4_EXPORT Platform_control
00047 : public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
00048 {
00049 public:
00051 enum Opcode
00052 {
00053 Suspend = L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00054 Shutdown = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00055 Cpu_enable = L4_PLATFORM_CTL_CPU_ENABLE_OP,
00056 Cpu_disable = L4_PLATFORM_CTL_CPU_DISABLE_OP
00057 };
00058
00065 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00066 l4_msgtag_t, system_suspend, (l4_umword_t extras));
00067
00073 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00074 l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00075
00083 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00084 l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00085
00093 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00094 l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00095
00096 typedef L4::Typeid::Rpcsys<system_suspend_t, system_shutdown_t,
00097 cpu_enable_t, cpu_disable_t> Rpcsys;
00098 };
00099
00100 }
00101

```

## 15.345 l4/sys/platform\_control.h File Reference

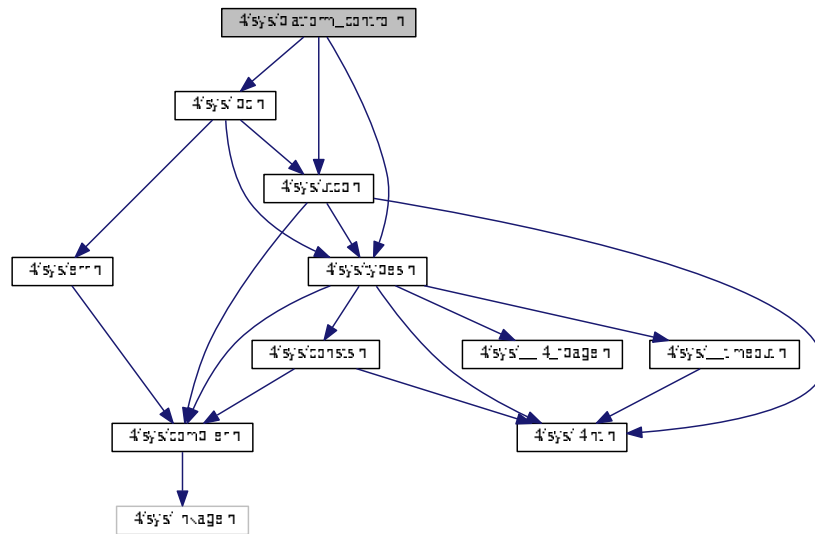
Platform control object.

```

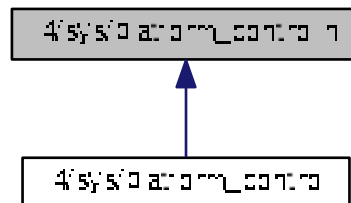
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for platform\_control.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_platform_ctl_ops` { `L4_PLATFORM_CTL_SYS_SUSPEND_OP` = 0UL, `L4_PLATFORM_CTL_SYS_SHUTDOWN_OP` = 1UL, `L4_PLATFORM_CTL_CPU_ENABLE_OP` = 3UL, `L4_PLATFORM_CTL_CPU_DISABLE_OP` = 4UL }
- Operations on platform-control objects.
- enum `L4_platform_ctl_proto` { `L4_PROTO_PLATFORM_CTL` = 0 }
- Predefined protocol type for messages to platform-control objects.

## Functions

- `l4_msgtag_t l4_platform_ctl_system_suspend(l4_cap_idx_t pfc, l4_umword_t extras) L4_NOTHROW`  
Enter suspend to RAM.

- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_system\\_shutdown](#) ([l4\\_cap\\_idx\\_t pfc](#), [l4\\_umword\\_t reboot](#)) [L4\\_NOTHROW](#)  
*Shutdown or reboot the system.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_enable](#) ([l4\\_cap\\_idx\\_t pfc](#), [l4\\_umword\\_t phys\\_id](#)) [L4\\_NOTHROW](#)  
*Enable an offline CPU.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_disable](#) ([l4\\_cap\\_idx\\_t pfc](#), [l4\\_umword\\_t phys\\_id](#)) [L4\\_NOTHROW](#)  
*Disable an online CPU.*

### 15.345.1 Detailed Description

Platform control object.

Definition in file [platform\\_control.h](#).

## 15.346 platform\_control.h

```

00001
00005 /*
00006 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025 #include <l4/sys/utcb.h>
00026
00051 L4_INLINE l4_msgtag_t
00052 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00053 l4_umword_t extras) L4_NOTHROW;
00054
00058 L4_INLINE l4_msgtag_t
00059 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00060 l4_umword_t extras,
00061 l4_utcb_t *utcb) L4_NOTHROW;
00062
00063
00072 L4_INLINE l4_msgtag_t
00073 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00074 l4_umword_t reboot) L4_NOTHROW;
00075
00079 L4_INLINE l4_msgtag_t
00080 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00081 l4_umword_t reboot,
00082 l4_utcb_t *utcb) L4_NOTHROW;
00083
00092 L4_INLINE l4_msgtag_t
00093 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00094 l4_umword_t phys_id) L4_NOTHROW;
00095
00099 L4_INLINE l4_msgtag_t
00100 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00101 l4_umword_t phys_id,
00102 l4_utcb_t *utcb) L4_NOTHROW;
00103
00112 L4_INLINE l4_msgtag_t
00113 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00114 l4_umword_t phys_id) L4_NOTHROW;
00115
00119 L4_INLINE l4_msgtag_t
00120 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,

```

```

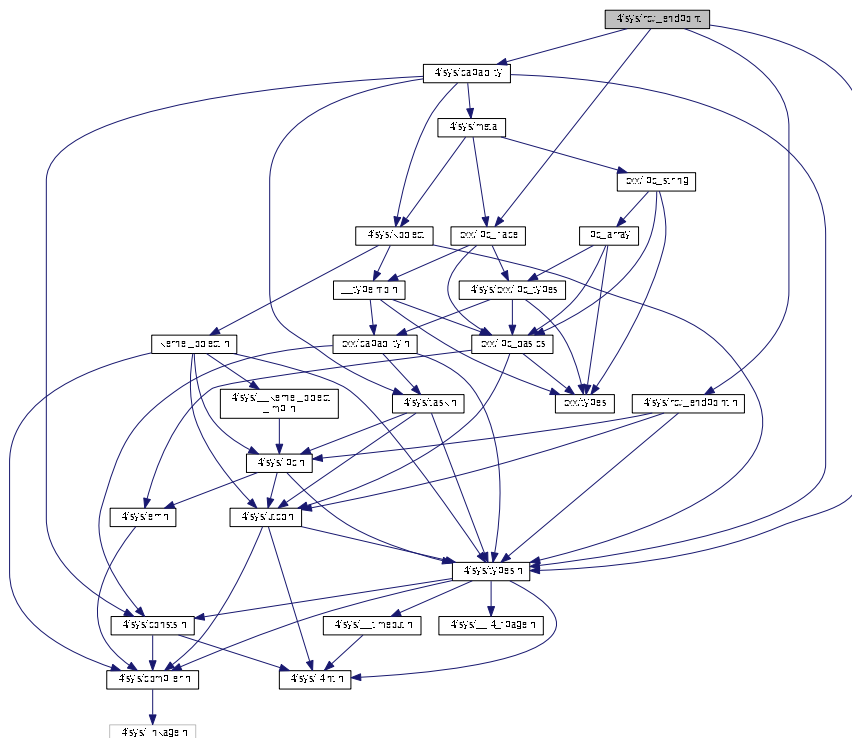
00121 l4_umword_t phys_id,
00122 l4_utcb_t *utcb) L4_NOTHROW;
00123 /* ends l4_platform_control_api group */
00124
00125
00126
00135 enum L4_platform_ctl_ops
00136 {
00137 L4_PLATFORM_CTL_SYS_SUSPEND_OP = 0UL,
00138 L4_PLATFORM_CTL_SYS_SHUTDOWN_OP = 1UL,
00139 L4_PLATFORM_CTL_CPU_ENABLE_OP = 3UL,
00140 L4_PLATFORM_CTL_CPU_DISABLE_OP = 4UL,
00141 };
00142
00147 enum L4_platform_ctl_proto
00148 {
00154 L4_PROTO_PLATFORM_CTL = 0
00155 };
00156
00157 /* IMPLEMENTATION -----*/
00158
00159 #include <l4/sys/ipc.h>
00160
00161 L4_INLINE l4_msgtag_t
00162 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00163 l4_umword_t extras,
00164 l4_utcb_t *utcb) L4_NOTHROW
00165 {
00166 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00167 v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
00168 v->mr[1] = extras;
00169 return l4_ipc_call(pfc, utcb, l4_msgtag(
00170 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00171 L4_IPC_NEVER);
00172 }
00173
00174 L4_INLINE l4_msgtag_t
00175 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00176 l4_umword_t reboot,
00177 l4_utcb_t *utcb) L4_NOTHROW
00178 {
00179 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00180 v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
00181 v->mr[1] = reboot;
00182 return l4_ipc_call(pfc, utcb, l4_msgtag(
00183 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00184 L4_IPC_NEVER);
00185 }
00186
00187 L4_INLINE l4_msgtag_t
00188 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00189 l4_umword_t extras) L4_NOTHROW
00190 {
00191 return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
00192 }
00193
00194 L4_INLINE l4_msgtag_t
00195 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00196 l4_umword_t reboot) L4_NOTHROW
00197 {
00198 return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
00199 }
00200
00201 L4_INLINE l4_msgtag_t
00202 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00203 l4_umword_t phys_id,
00204 l4_utcb_t *utcb) L4_NOTHROW
00205 {
00206 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00207 v->mr[0] = L4_PLATFORM_CTL_CPU_ENABLE_OP;
00208 v->mr[1] = phys_id;
00209 return l4_ipc_call(pfc, utcb, l4_msgtag(
00210 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00211 L4_IPC_NEVER);
00212 }
00213
00214 L4_INLINE l4_msgtag_t
00215 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00216 l4_umword_t phys_id,
00217 l4_utcb_t *utcb) L4_NOTHROW
00218 {
00219 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00220 v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
00221 v->mr[1] = phys_id;
00222 return l4_ipc_call(pfc, utcb, l4_msgtag(
00223 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00224 L4_IPC_NEVER);

```

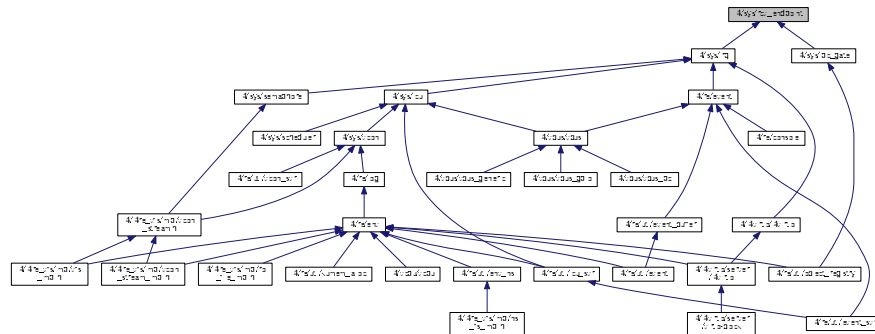


### 15.347 I4/sys/rcv\_endpoint File Reference

```
#include <l4/sys/rcv_endpoint.h>
#include <l4/sys/types.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for rcv_endpoint:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Rcv\\_endpoint](#)  
*Interface for kernel objects that allow to receive IPC from them.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 15.347.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file [rcv\\_endpoint](#).

## 15.348 rcv\_endpoint

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019
00020 #include <l4/sys/rcv_endpoint.h>
00021 #include <l4/sys/types.h>
00022 #include <l4/sys/capability>
00023 #include <l4/sys/cxx/ipc_iface>
00024
00025 namespace L4 {

```

```

00030
00031 class Thread;
00032
00040 class L4_EXPORT Rcv_endpoint :
00041 public Kobject_t<Rcv_endpoint, Kobject, L4_PROTO_KOBJECT,
00042 Type_info::Demand_t<1> >
00043 {
00044 public:
00059 L4_INLINE_RPC_OP(L4_RCV_EP_BIND_OP,
00060 l4_msgtag_t, bind_thread, (Ipc::Opt<Ipc::Cap<Thread> > t,
00061 l4_umword_t label));
00061
00062 typedef L4::Typeid::Rpcs_sys<bind_thread_t>
00063 Rpcs;
00064 };
00065 }

```

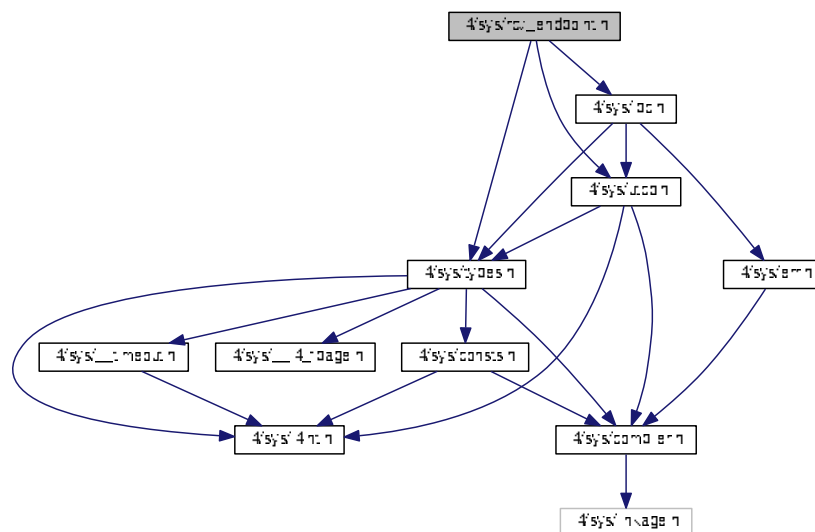
## 15.349 l4/sys/rcv\_endpoint.h File Reference

Receive endpoint C interface.

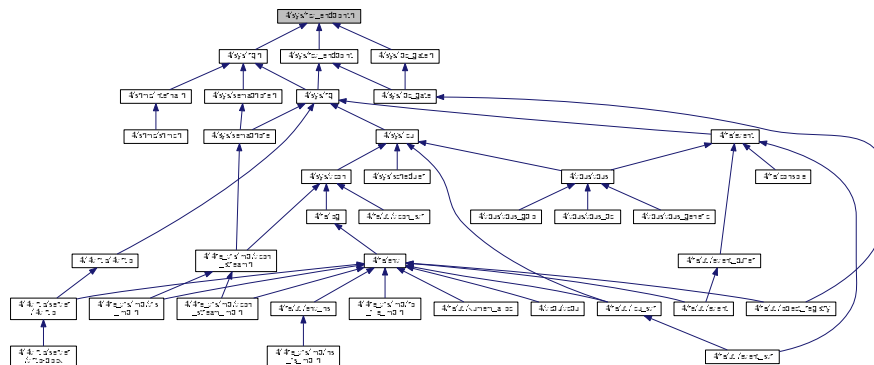
```

#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/ipc.h>
Include dependency graph for rcv_endpoint.h:

```



This graph shows which files directly or indirectly include this file:



## Enumerations

- `enum L4_rcv_ep_ops { L4_RCV_EP_BIND_OP = 0x10 }`  
Receive endpoint operations.

## Functions

- `l4_msgtag_t l4_rcv_ep_bind_thread (l4_cap_idx_t ep, l4_cap_idx_t thread, l4_umword_t label)`  
Bind the IPC gate to a thread.

### 15.349.1 Detailed Description

Receive endpoint C interface.

Definition in file [rcv\\_endpoint.h](#).

### 15.349.2 Enumeration Type Documentation

#### 15.349.2.1 L4\_rcv\_ep\_ops

`enum L4_rcv_ep_ops`

Receive endpoint operations.

#### Enumerator

|                   |                 |
|-------------------|-----------------|
| L4_RCV_EP_BIND_OP | Bind operation. |
|-------------------|-----------------|

Definition at line 56 of file [rcv\\_endpoint.h](#).

## 15.350 rcv\_endpoint.h

```

00001
00005 /*
00006 * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 #include <l4/sys/utcb.h>
00024 #include <l4/sys/types.h>
00025
00043 L4_INLINE l4_msgtag_t
00044 l4_rcv_ep_bind_thread(l4_cap_idx_t ep,
00045 l4_cap_idx_t thread,
00046 l4_umword_t label);
00051 L4_INLINE l4_msgtag_t
00052 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep, l4_cap_idx_t thread,
00053 l4_umword_t label, l4_utcb_t *utcb);
00054
00056 enum L4_rcv_ep_ops
00057 {
00058 L4_RCV_EP_BIND_OP = 0x10,
00059 };
00060
00061 /* IMPLEMENTATION -----*/
00062
00063 #include <l4/sys/ipc.h>
00064
00065 L4_INLINE l4_msgtag_t
00066 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep,
00067 l4_cap_idx_t thread, l4_umword_t label,
00068 l4_utcb_t *utcb)
00069 {
00070 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00071 m->mr[0] = L4_RCV_EP_BIND_OP;
00072 m->mr[1] = label;
00073 m->mr[2] = l4_map_obj_control(0, 0);
00074 m->mr[3] = l4_obj_fpage(thread, 0, L4_FPAGE_RWX).raw;
00075 return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00076 L4_IPC_NEVER);
00077 }
00078
00079 L4_INLINE l4_msgtag_t
00080 l4_rcv_ep_bind_thread(l4_cap_idx_t ep,
00081 l4_cap_idx_t thread,
00082 l4_umword_t label)
00083 {
00084 return l4_rcv_ep_bind_thread_u(ep, thread, label, l4_utcb());
00085 }
00086

```

## 15.351 l4/sys/scheduler File Reference

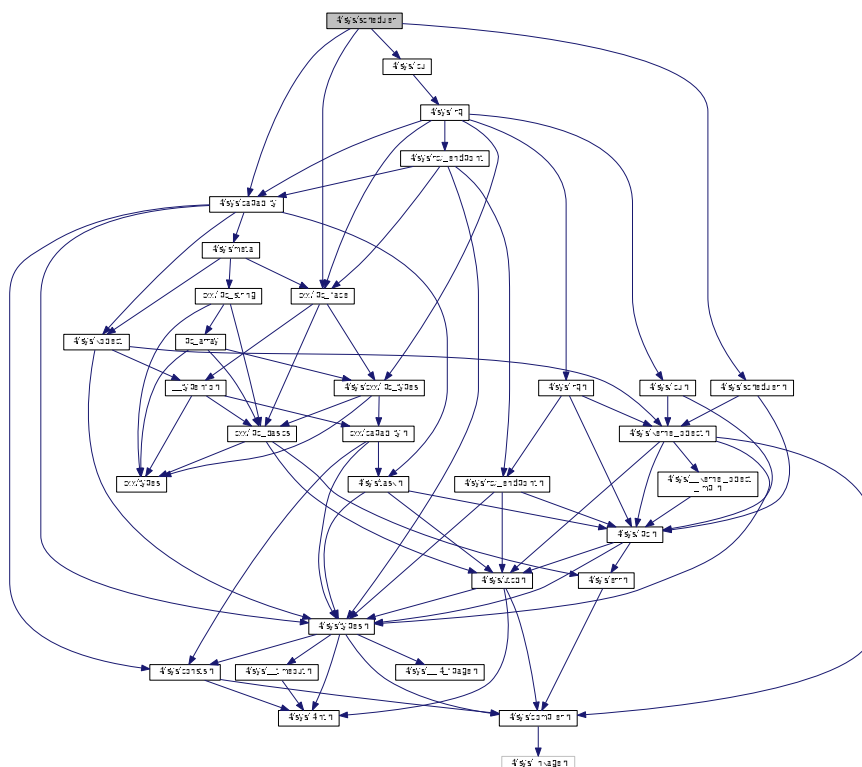
Scheduler object functions.

```

#include <l4/sys/icu>
#include <l4/sys/scheduler.h>

```

```
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for scheduler:
```



## Data Structures

- class [L4::Scheduler](#)

*C++ interface of the [Scheduler](#) kernel object.*

## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

### 15.351.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler](#).

## 15.352 scheduler

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/icu>
00027 #include <l4/sys/scheduler.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4 {
00032
00042 class L4_EXPORT Scheduler :
00043 public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00044 Type_info::Demand_t<1> >
00045 {
00046 public:
00047 // ABI function for 'info' call
00048 L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00049 l4_msgtag_t, info, (l4_umword_t gran_offset,
00050 l4_umword_t *map,
00051 l4_umword_t *cpu_max));
00052
00066 l4_msgtag_t info(l4_umword_t *cpu_max,
00067 l4_sched_cpu_set_t *cpus,
00068 l4_utcb_t *utcb = l4_utcb()) const throw()
00069 {
00070 l4_umword_t max = 0;
00071 l4_msgtag_t t =
00072 info_t::call(c(), cpus->gran_offset, &cpus->map, &max, utcb);
00073 if (cpu_max) *cpu_max = max;
00074 return t;
00075 }
00097 L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00098 l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread,
00099 l4_sched_param_t const &sp));
00126 L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00127 l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00128 l4_kernel_clock_t *us));
00139 bool is_online(l4_umword_t cpu, l4_utcb_t *utcb =
00140 l4_utcb()) const throw()
00141 { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00142 typedef L4::Typeid::Rpcsys<info_t, run_thread_t, idle_time_t>
00143 Rpc;
00144 }

```

## 15.353 l4/sys/scheduler.h File Reference

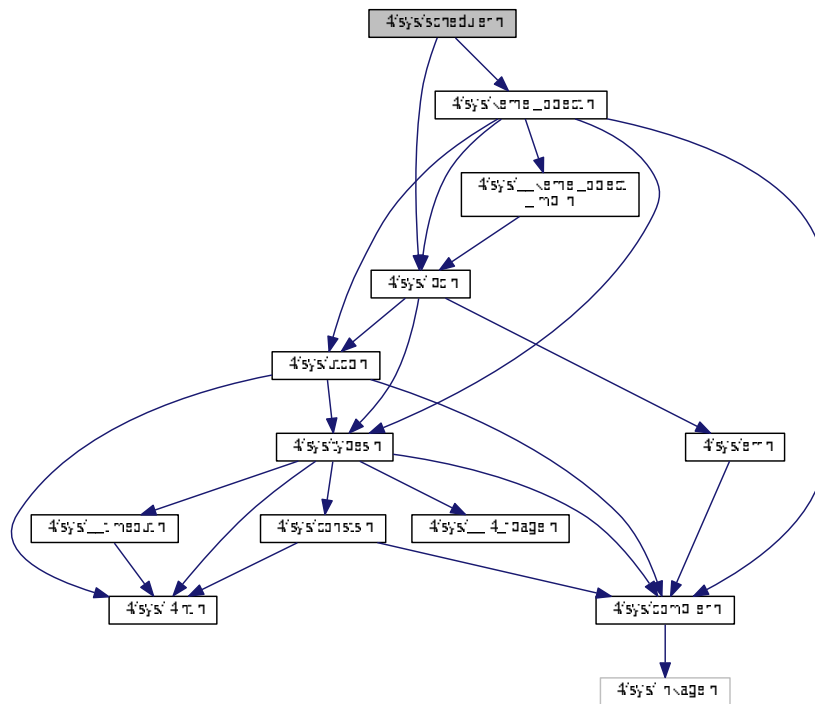
Scheduler object functions.

```

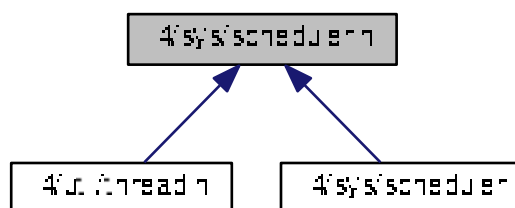
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for scheduler.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_sched\\_cpu\\_set\\_t](#)  
*CPU sets.*
- struct [l4\\_sched\\_param\\_t](#)  
*Scheduler parameter set.*



## Typedefs

- typedef struct `l4_sched_cpu_set_t` `l4_sched_cpu_set_t`  
*CPU sets.*
- typedef struct `l4_sched_param_t` `l4_sched_param_t`  
*Scheduler parameter set.*

## Enumerations

- enum `L4_scheduler_ops` { `L4_SCHEDULER_INFO_OP` = 0UL, `L4_SCHEDULER_RUN_THREAD_OP` = 1UL, `L4_SCHEDULER_IDLE_TIME_OP` = 2UL }
- Operations on the Scheduler object.*

## Functions

- `l4_sched_cpu_set_t` `l4_sched_cpu_set` (`l4_umword_t` offset, unsigned char granularity, `l4_umword_t` map=1) `L4_NOTHROW`
- `l4_msgtag_t` `l4_scheduler_info` (`l4_cap_idx_t` scheduler, `l4_umword_t` \*cpu\_max, `l4_sched_cpu_set_t` \*cpus) `L4_NOTHROW`  
*Get scheduler information.*
- `l4_sched_param_t` `l4_sched_param` (unsigned prio, `l4_cpu_time_t` quantum=0) `L4_NOTHROW`  
*Construct scheduler parameter.*
- `l4_msgtag_t` `l4_scheduler_run_thread` (`l4_cap_idx_t` scheduler, `l4_cap_idx_t` thread, `l4_sched_param_t` const \*sp) `L4_NOTHROW`  
*Run a thread on a Scheduler.*
- `l4_msgtag_t` `l4_scheduler_idle_time` (`l4_cap_idx_t` scheduler, `l4_sched_cpu_set_t` const \*cpus, `l4_kernel_clock_t` \*us) `L4_NOTHROW`  
*Query the idle time (in  $\mu$ s) of a CPU.*
- int `l4_scheduler_is_online` (`l4_cap_idx_t` scheduler, `l4_umword_t` cpu) `L4_NOTHROW`  
*Query if a CPU is online.*

### 15.353.1 Detailed Description

Scheduler object functions.

Definition in file `scheduler.h`.

## 15.354 scheduler.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
```

```

00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/sys/kernel_object.h>
00026 #include <l4/sys/ipc.h>
00027
00044 typedef struct l4_sched_cpu_set_t
00045 {
00058 l4_umword_t gran_offset;
00059
00063 l4_umword_t map;
00064
00065 #ifdef __cplusplus
00066 unsigned char granularity() const { return gran_offset >> 24; }
00069 unsigned offset() const { return gran_offset & 0x00ffffff; }
00071 void set(unsigned char granularity, unsigned offset)
00072 { gran_offset = ((l4_umword_t)granularity << 24) | (offset & 0x00ffffff); }
00073 #endif
00074 } l4_sched_cpu_set_t;
00075
00086 L4_INLINE l4_sched_cpu_set_t
00087 l4_sched_cpu_set(l4_umword_t offset, unsigned char
00088 granularity,
00089 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;
00104 L4_INLINE l4_msgtag_t
00105 l4_scheduler_info(l4_cap_idx_t scheduler,
00106 l4_umword_t *cpu_max,
00107 l4_sched_cpu_set_t *cpus) L4_NOTHROW;
00111 L4_INLINE l4_msgtag_t
00112 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00113 l4_sched_cpu_set_t *cpus, l4_utcb_t *utcb)
00114 L4_NOTHROW;
00115
00120 typedef struct l4_sched_param_t
00121 {
00122 l4_sched_cpu_set_t affinity;
00123 l4_umword_t prio;
00124 l4_umword_t quantum;
00125 } l4_sched_param_t;
00126
00131 L4_INLINE l4_sched_param_t
00132 l4_sched_param(unsigned prio,
00133 l4_cpu_time_t quantum L4_DEFAULT_PARAM(0))
00134 L4_NOTHROW;
00142 L4_INLINE l4_msgtag_t
00143 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00144 l4_cap_idx_t thread, l4_sched_param_t const *sp)
00145 L4_NOTHROW;
00149 L4_INLINE l4_msgtag_t
00150 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00151 l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW;
00160 L4_INLINE l4_msgtag_t
00161 l4_scheduler_idle_time(l4_cap_idx_t scheduler,
00162 l4_sched_cpu_set_t const *cpus,
00163 l4_kernel_clock_t *us) L4_NOTHROW;
00167 L4_INLINE l4_msgtag_t
00168 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *
00169 cpus,
00170 l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW;
00171
00172
00183 L4_INLINE int
00184 l4_scheduler_is_online(l4_cap_idx_t scheduler,
00185 l4_umword_t cpu) L4_NOTHROW;
00189 L4_INLINE int
00190 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00191 l4_utcb_t *utcb) L4_NOTHROW;
00192
00193
00201 enum l4_scheduler_ops
00202 {
00203 L4_SCHEDULER_INFO_OP = 0UL,
00204 L4_SCHEDULER_RUN_THREAD_OP = 1UL,
00205 L4_SCHEDULER_IDLE_TIME_OP = 2UL,

```

```

00206 };
00207
00208 /***** Implementations *****/
00209
00210 L4_INLINE l4_sched_cpu_set_t
00211 l4_sched_cpu_set(l4_umword_t offset, unsigned char
granularity,
00212 l4_umword_t map) L4_NOTHROW
00213 {
00214 l4_sched_cpu_set_t cs;
00215 cs.gran_offset = ((l4_umword_t)granularity << 24) |
offset;
00216 cs.map = map;
00217 return cs;
00218 }
00219
00220 L4_INLINE l4_sched_param_t
00221 l4_sched_param(unsigned prio, l4_cpu_time_t quantum)
L4_NOTHROW
00222 {
00223 l4_sched_param_t sp;
00224 sp.prio = prio;
00225 sp.quantum = quantum;
00226 sp.affinity = l4_sched_cpu_set(0, ~0, 1);
00227 return sp;
00228 }
00229
00230
00231 L4_INLINE l4_msgtag_t
00232 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00233 l4_sched_cpu_set_t *cpus, l4_utcb_t *utcb)
L4_NOTHROW
00234 {
00235 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00236 l4_msgtag_t res;
00237
00238 m->mr[0] = L4_SCHEDULER_INFO_OP;
00239 m->mr[1] = cpus->gran_offset;
00240
00241 res = l4_ipc_call(scheduler, utcb, l4_msgtag(
L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);
00242
00243 if (l4_msgtag_has_error(res))
00244 return res;
00245
00246 cpus->map = m->mr[0];
00247
00248 if (cpu_max)
00249 *cpu_max = m->mr[1];
00250
00251 return res;
00252 }
00253
00254 L4_INLINE l4_msgtag_t
00255 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00256 l4_sched_param_t const *sp, l4_utcb_t *utcb)
L4_NOTHROW
00257 {
00258 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00259 m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
00260 m->mr[1] = sp->affinity.gran_offset;
00261 m->mr[2] = sp->affinity.map;
00262 m->mr[3] = sp->prio;
00263 m->mr[4] = sp->quantum;
00264 m->mr[5] = l4_map_obj_control(0, 0);
00265 m->mr[6] = l4_obj_fpage(thread, 0, L4_FPAGE_RWX).raw;
00266
00267 return l4_ipc_call(scheduler, utcb, l4_msgtag(
L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
00268 }
00269
00270 L4_INLINE l4_msgtag_t
00271 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *
cpus,
00272 l4_kernel_clock_t *us, l4_utcb_t *utcb)
L4_NOTHROW
00273 {
00274 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00275 l4_msgtag_t res;
00276
00277 v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
00278 v->mr[1] = cpus->gran_offset;
00279 v->mr[2] = cpus->map;
00280
00281 res = l4_ipc_call(scheduler, utcb,
00282 l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0),
L4_IPC_NEVER);

```

```

00283
00284 if (l4_msgtag_has_error(res))
00285 return res;
00286
00287 *us = v->mr64[l4_utcb_mr64_idx(0)];
00288
00289 return res;
00290 }
00291
00292
00293 L4_INLINE int
00294 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00295 l4_utcb_t *utcb) L4_NOTHROW
00296 {
00297 l4_sched_cpu_set_t s;
00298 l4_msgtag_t r;
00299 s.gran_offset = cpu;
00300 r = l4_scheduler_info_u(scheduler, NULL, &s, utcb);
00301 if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
00302 return 0;
00303
00304 return s.map & 1;
00305 }
00306
00307
00308 L4_INLINE l4_msgtag_t
00309 l4_scheduler_info(l4_cap_idx_t scheduler,
00310 l4_umword_t *cpu_max,
00311 l4_sched_cpu_set_t *cpus) L4_NOTHROW
00312 {
00313 return l4_scheduler_info_u(scheduler, cpu_max, cpus, l4_utcb());
00314 }
00315
00316 L4_INLINE l4_msgtag_t
00317 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00318 l4_cap_idx_t thread, l4_sched_param_t const *sp)
00319 L4_NOTHROW
00320 {
00321 return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
00322 }
00323
00324 L4_INLINE l4_msgtag_t
00325 l4_scheduler_idle_time(l4_cap_idx_t scheduler,
00326 l4_sched_cpu_set_t const *cpus,
00327 l4_kernel_clock_t *us) L4_NOTHROW
00328 {
00329 return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
00330 }
00331
00332 L4_INLINE int
00333 l4_scheduler_is_online(l4_cap_idx_t scheduler,
00334 l4_umword_t cpu) L4_NOTHROW
00335 {
00336 return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
00337 }

```

## 15.355 l4/sys/semaphore File Reference

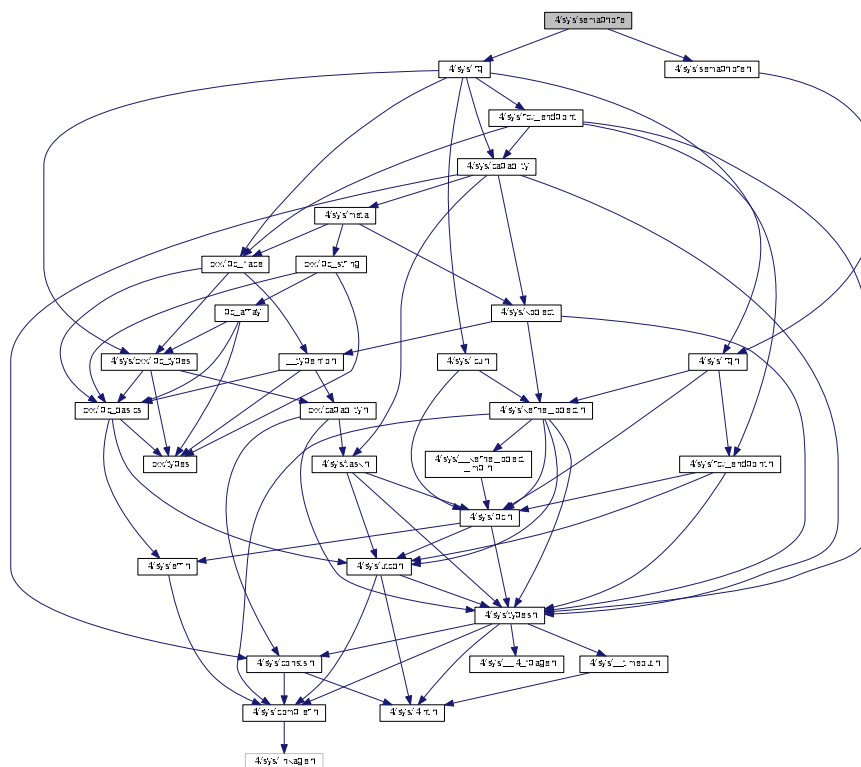
Semaphore class definition.

```

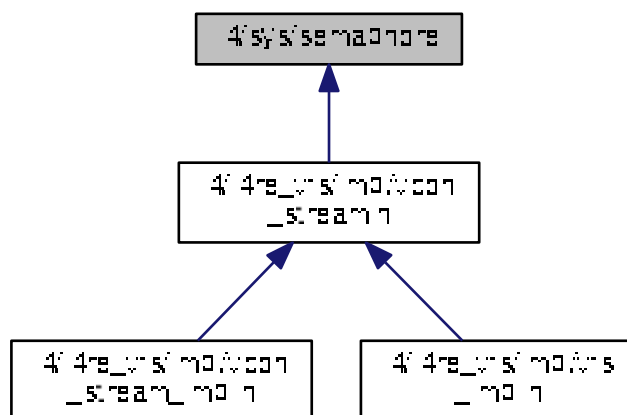
#include <l4/sys/irq>
#include <l4/sys/semaphore.h>

```

Include dependency graph for semaphore:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4::Semaphore](#)  
*Kernel-provided semaphore object.*

## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

### 15.355.1 Detailed Description

Semaphore class definition.

Definition in file [semaphore](#).

## 15.356 semaphore

```

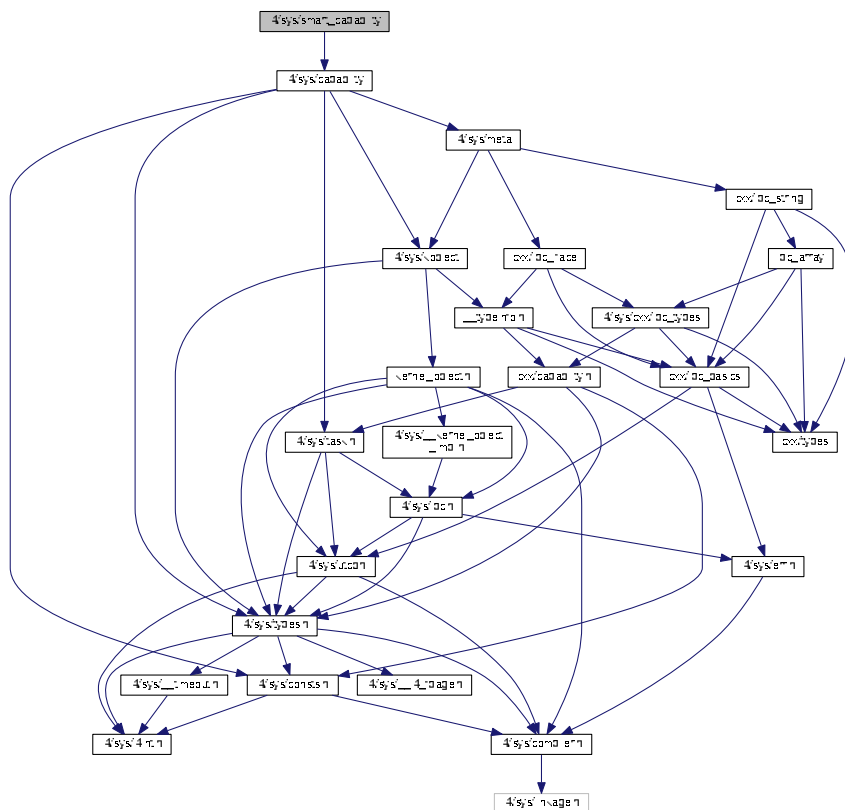
00001 // vi:set ft=c++: -- Mode: C++ --
00006 /*
00007 * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/irq>
00026 #include <l4/sys/semaphore.h>
00027
00028 namespace L4 {
00029
00051 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00052 {
00065 l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) throw()
00066 { return trigger(utcb); }
00067
00084 l4_msgtag_t down(l4_timeout_t timeout =
00085 L4_IPC_NEVER,
00086 l4_utcb_t *utcb = l4_utcb()) throw()
00087 { return l4_semaphore_down_u(cap(), timeout, utcb); }
00088 };
00089

```

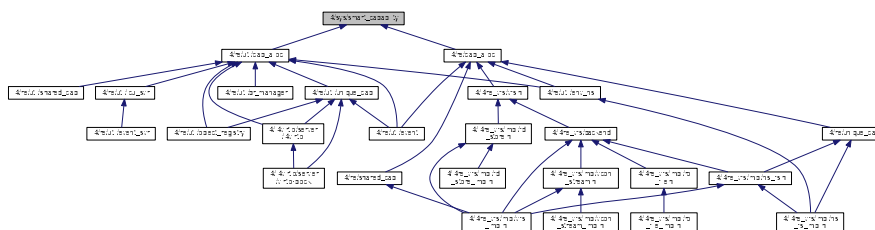
## 15.357 l4/sys/smart\_capability File Reference

L4::Capability class.

Include dependency graph for smart\_capability:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Smart_cap< T, SMART >`  
*Smart capability class.*

## Namespaces

- L4  
*L4 low-level kernel interface.*

## Functions

- `template<typename T, typename F, typename SMART >`  
`Smart_cap< T, SMART > L4::cap_cast (Smart_cap< F, SMART > const &c) throw ()`  
*static\_cast for (smart) capabilities.*
- `template<typename T, typename F, typename SMART >`  
`Smart_cap< T, SMART > L4::cap_reinterpret_cast (Smart_cap< F, SMART > const &c) throw ()`  
*reinterpret\_cast for (smart) capabilities.*

### 15.357.1 Detailed Description

L4::Capability class.

#### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [smart\\_capability](#).

## 15.358 smart\_capability

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010 * (c) 2008-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00032 template< typename T, typename SMART >
00033 class Smart_cap : public Cap_base, private SMART
00034 {
00035 public:
00036 SMART const &smart() const { return *this; }
00037
00038 void _delete() throw()
00039 {
00040 SMART::free(const_cast<Smart_cap<T, SMART>&>(*this));
00041 }
00042
00043 Cap<T> release() const throw()
00044 {
00045 l4_cap_idx_t r = cap();
00046 SMART::invalidate(const_cast<Smart_cap<T, SMART>&>(*this));
00047
00048 return Cap<T>(r);
00049 }
00050
00051 void reset()
00052 {
00053 _c = L4_INVALID_CAP;
00054 }
00055
00056 void reset()
00057 {
00058 _c = L4_INVALID_CAP;
00059 }
00060
00061 void reset()
00062 {
00063 _c = L4_INVALID_CAP;
00064 }
00065
00066 void reset()
00067 {
00068 _c = L4_INVALID_CAP;
00069 }
00070
00071 void reset()
00072 {
00073 _c = L4_INVALID_CAP;
00074 }
00075
00076 void reset()
00077 {
00078 _c = L4_INVALID_CAP;
00079 }
00080
00081 void reset()
00082 {
00083 _c = L4_INVALID_CAP;
00084 }
00085
00086 void reset()
00087 {
00088 _c = L4_INVALID_CAP;
00089 }
00090
00091 void reset()
00092 {
00093 _c = L4_INVALID_CAP;
00094 }
00095
00096 void reset()
00097 {
00098 _c = L4_INVALID_CAP;
00099 }
00100
00101 void reset()
00102 {
00103 _c = L4_INVALID_CAP;
00104 }
00105
00106 void reset()
00107 {
00108 _c = L4_INVALID_CAP;
00109 }
00110
00111 void reset()
00112 {
00113 _c = L4_INVALID_CAP;
00114 }
00115
00116 void reset()
00117 {
00118 _c = L4_INVALID_CAP;
00119 }
00120
00121 void reset()
00122 {
00123 _c = L4_INVALID_CAP;
00124 }
00125
00126 void reset()
00127 {
00128 _c = L4_INVALID_CAP;
00129 }
00130
00131 void reset()
00132 {
00133 _c = L4_INVALID_CAP;
00134 }
00135
00136 void reset()
00137 {
00138 _c = L4_INVALID_CAP;
00139 }
00140
00141 void reset()
00142 {
00143 _c = L4_INVALID_CAP;
00144 }
00145
00146 void reset()
00147 {
00148 _c = L4_INVALID_CAP;
00149 }
00150
00151 void reset()
00152 {
00153 _c = L4_INVALID_CAP;
00154 }
00155
00156 void reset()
00157 {
00158 _c = L4_INVALID_CAP;
00159 }
00160
00161 void reset()
00162 {
00163 _c = L4_INVALID_CAP;
00164 }
00165
00166 void reset()
00167 {
00168 _c = L4_INVALID_CAP;
00169 }
00170
00171 void reset()
00172 {
00173 _c = L4_INVALID_CAP;
00174 }
00175
00176 void reset()
00177 {
00178 _c = L4_INVALID_CAP;
00179 }
00180
00181 void reset()
00182 {
00183 _c = L4_INVALID_CAP;
00184 }
00185
00186 void reset()
00187 {
00188 _c = L4_INVALID_CAP;
00189 }
00190
00191 void reset()
00192 {
00193 _c = L4_INVALID_CAP;
00194 }
00195
00196 void reset()
00197 {
00198 _c = L4_INVALID_CAP;
00199 }
00200
00201 void reset()
00202 {
00203 _c = L4_INVALID_CAP;
00204 }
00205
00206 void reset()
00207 {
00208 _c = L4_INVALID_CAP;
00209 }
00210
00211 void reset()
00212 {
00213 _c = L4_INVALID_CAP;
00214 }
00215
00216 void reset()
00217 {
00218 _c = L4_INVALID_CAP;
00219 }
00220
00221 void reset()
00222 {
00223 _c = L4_INVALID_CAP;
00224 }
00225
00226 void reset()
00227 {
00228 _c = L4_INVALID_CAP;
00229 }
00230
00231 void reset()
00232 {
00233 _c = L4_INVALID_CAP;
00234 }
00235
00236 void reset()
00237 {
00238 _c = L4_INVALID_CAP;
00239 }
00240
00241 void reset()
00242 {
00243 _c = L4_INVALID_CAP;
00244 }
00245
00246 void reset()
00247 {
00248 _c = L4_INVALID_CAP;
00249 }
00250
00251 void reset()
00252 {
00253 _c = L4_INVALID_CAP;
00254 }
00255
00256 void reset()
00257 {
00258 _c = L4_INVALID_CAP;
00259 }
00260
00261 void reset()
00262 {
00263 _c = L4_INVALID_CAP;
00264 }
00265
00266 void reset()
00267 {
00268 _c = L4_INVALID_CAP;
00269 }
00270
00271 void reset()
00272 {
00273 _c = L4_INVALID_CAP;
00274 }
00275
00276 void reset()
00277 {
00278 _c = L4_INVALID_CAP;
00279 }
00280
00281 void reset()
00282 {
00283 _c = L4_INVALID_CAP;
00284 }
00285
00286 void reset()
00287 {
00288 _c = L4_INVALID_CAP;
00289 }
00290
00291 void reset()
00292 {
00293 _c = L4_INVALID_CAP;
00294 }
00295
00296 void reset()
00297 {
00298 _c = L4_INVALID_CAP;
00299 }
00300
00301 void reset()
00302 {
00303 _c = L4_INVALID_CAP;
00304 }
00305
00306 void reset()
00307 {
00308 _c = L4_INVALID_CAP;
00309 }
00310
00311 void reset()
00312 {
00313 _c = L4_INVALID_CAP;
00314 }
00315
00316 void reset()
00317 {
00318 _c = L4_INVALID_CAP;
00319 }
00320
00321 void reset()
00322 {
00323 _c = L4_INVALID_CAP;
00324 }
00325
00326 void reset()
00327 {
00328 _c = L4_INVALID_CAP;
00329 }
00330
00331 void reset()
00332 {
00333 _c = L4_INVALID_CAP;
00334 }
00335
00336 void reset()
00337 {
00338 _c = L4_INVALID_CAP;
00339 }
00340
00341 void reset()
00342 {
00343 _c = L4_INVALID_CAP;
00344 }
00345
00346 void reset()
00347 {
00348 _c = L4_INVALID_CAP;
00349 }
00350
00351 void reset()
00352 {
00353 _c = L4_INVALID_CAP;
00354 }
00355
00356 void reset()
00357 {
00358 _c = L4_INVALID_CAP;
00359 }
00360
00361 void reset()
00362 {
00363 _c = L4_INVALID_CAP;
00364 }
00365
00366 void reset()
00367 {
00368 _c = L4_INVALID_CAP;
00369 }
00370
00371 void reset()
00372 {
00373 _c = L4_INVALID_CAP;
00374 }
00375
00376 void reset()
00377 {
00378 _c = L4_INVALID_CAP;
00379 }
00380
00381 void reset()
00382 {
00383 _c = L4_INVALID_CAP;
00384 }
00385
00386 void reset()
00387 {
00388 _c = L4_INVALID_CAP;
00389 }
00390
00391 void reset()
00392 {
00393 _c = L4_INVALID_CAP;
00394 }
00395
00396 void reset()
00397 {
00398 _c = L4_INVALID_CAP;
00399 }
00400
00401 void reset()
00402 {
00403 _c = L4_INVALID_CAP;
00404 }
00405
00406 void reset()
00407 {
00408 _c = L4_INVALID_CAP;
00409 }
00410
00411 void reset()
00412 {
00413 _c = L4_INVALID_CAP;
00414 }
00415
00416 void reset()
00417 {
00418 _c = L4_INVALID_CAP;
00419 }
00420
00421 void reset()
00422 {
00423 _c = L4_INVALID_CAP;
00424 }
00425
00426 void reset()
00427 {
00428 _c = L4_INVALID_CAP;
00429 }
00430
00431 void reset()
00432 {
00433 _c = L4_INVALID_CAP;
00434 }
00435
00436 void reset()
00437 {
00438 _c = L4_INVALID_CAP;
00439 }
00440
00441 void reset()
00442 {
00443 _c = L4_INVALID_CAP;
00444 }
00445
00446 void reset()
00447 {
00448 _c = L4_INVALID_CAP;
00449 }
00450
00451 void reset()
00452 {
00453 _c = L4_INVALID_CAP;
00454 }
00455
00456 void reset()
00457 {
00458 _c = L4_INVALID_CAP;
00459 }
00460
00461 void reset()
00462 {
00463 _c = L4_INVALID_CAP;
00464 }
00465
00466 void reset()
00467 {
00468 _c = L4_INVALID_CAP;
00469 }
00470
00471 void reset()
00472 {
00473 _c = L4_INVALID_CAP;
00474 }
00475
00476 void reset()
00477 {
00478 _c = L4_INVALID_CAP;
00479 }
00480
00481 void reset()
00482 {
00483 _c = L4_INVALID_CAP;
00484 }
00485
00486 void reset()
00487 {
00488 _c = L4_INVALID_CAP;
00489 }
00490
00491 void reset()
00492 {
00493 _c = L4_INVALID_CAP;
00494 }
00495
00496 void reset()
00497 {
00498 _c = L4_INVALID_CAP;
00499 }
00500
00501 void reset()
00502 {
00503 _c = L4_INVALID_CAP;
00504 }
00505
00506 void reset()
00507 {
00508 _c = L4_INVALID_CAP;
00509 }
00510
00511 void reset()
00512 {
00513 _c = L4_INVALID_CAP;
00514 }
00515
00516 void reset()
00517 {
00518 _c = L4_INVALID_CAP;
00519 }
00520
00521 void reset()
00522 {
00523 _c = L4_INVALID_CAP;
00524 }
00525
00526 void reset()
00527 {
00528 _c = L4_INVALID_CAP;
00529 }
00530
00531 void reset()
00532 {
00533 _c = L4_INVALID_CAP;
00534 }
00535
00536 void reset()
00537 {
00538 _c = L4_INVALID_CAP;
00539 }
00540
00541 void reset()
00542 {
00543 _c = L4_INVALID_CAP;
00544 }
00545
00546 void reset()
00547 {
00548 _c = L4_INVALID_CAP;
00549 }
00550
00551 void reset()
00552 {
00553 _c = L4_INVALID_CAP;
00554 }
00555
00556 void reset()
00557 {
00558 _c = L4_INVALID_CAP;
00559 }
00560
00561 void reset()
00562 {
00563 _c = L4_INVALID_CAP;
00564 }
00565
00566 void reset()
00567 {
00568 _c = L4_INVALID_CAP;
00569 }
00570
00571 void reset()
00572 {
00573 _c = L4_INVALID_CAP;
00574 }
00575
00576 void reset()
00577 {
00578 _c = L4_INVALID_CAP;
00579 }
00580
00581 void reset()
00582 {
00583 _c = L4_INVALID_CAP;
00584 }
00585
00586 void reset()
00587 {
00588 _c = L4_INVALID_CAP;
00589 }
00590
00591 void reset()
00592 {
00593 _c = L4_INVALID_CAP;
00594 }
00595
00596 void reset()
00597 {
00598 _c = L4_INVALID_CAP;
00599 }
00600
00601 void reset()
00602 {
00603 _c = L4_INVALID_CAP;
00604 }
00605
00606 void reset()
00607 {
00608 _c = L4_INVALID_CAP;
00609 }
00610
00611 void reset()
00612 {
00613 _c = L4_INVALID_CAP;
00614 }
00615
00616 void reset()
00617 {
00618 _c = L4_INVALID_CAP;
00619 }
00620
00621 void reset()
00622 {
00623 _c = L4_INVALID_CAP;
00624 }
00625
00626 void reset()
00627 {
00628 _c = L4_INVALID_CAP;
00629 }
00630
00631 void reset()
00632 {
00633 _c = L4_INVALID_CAP;
00634 }
00635
00636 void reset()
00637 {
00638 _c = L4_INVALID_CAP;
00639 }
00640
00641 void reset()
00642 {
00643 _c = L4_INVALID_CAP;
00644 }
00645
00646 void reset()
00647 {
00648 _c = L4_INVALID_CAP;
00649 }
00650
00651 void reset()
00652 {
00653 _c = L4_INVALID_CAP;
00654 }
00655
00656 void reset()
00657 {
00658 _c = L4_INVALID_CAP;
00659 }
00660
00661 void reset()
00662 {
00663 _c = L4_INVALID_CAP;
00664 }
00665
00666 void reset()
00667 {
00668 _c = L4_INVALID_CAP;
00669 }
00670
00671 void reset()
00672 {
00673 _c = L4_INVALID_CAP;
00674 }
00675
00676 void reset()
00677 {
00678 _c = L4_INVALID_CAP;
00679 }
00680
00681 void reset()
00682 {
00683 _c = L4_INVALID_CAP;
00684 }
00685
00686 void reset()
00687 {
00688 _c = L4_INVALID_CAP;
00689 }
00690
00691 void reset()
00692 {
00693 _c = L4_INVALID_CAP;
00694 }
00695
00696 void reset()
00697 {
00698 _c = L4_INVALID_CAP;
00699 }
00700
00701 void reset()
00702 {
00703 _c = L4_INVALID_CAP;
00704 }
00705
00706 void reset()
00707 {
00708 _c = L4_INVALID_CAP;
00709 }
00710
00711 void reset()
00712 {
00713 _c = L4_INVALID_CAP;
00714 }
00715
00716 void reset()
00717 {
00718 _c = L4_INVALID_CAP;
00719 }
00720
00721 void reset()
00722 {
00723 _c = L4_INVALID_CAP;
00724 }
00725
00726 void reset()
00727 {
00728 _c = L4_INVALID_CAP;
00729 }
00730
00731 void reset()
00732 {
00733 _c = L4_INVALID_CAP;
00734 }
00735
00736 void reset()
00737 {
00738 _c = L4_INVALID_CAP;
00739 }
00740
00741 void reset()
00742 {
00743 _c = L4_INVALID_CAP;
00744 }
00745
00746 void reset()
00747 {
00748 _c = L4_INVALID_CAP;
00749 }
00750
00751 void reset()
00752 {
00753 _c = L4_INVALID_CAP;
00754 }
00755
00756 void reset()
00757 {
00758 _c = L4_INVALID_CAP;
00759 }
00760
00761 void reset()
00762 {
00763 _c = L4_INVALID_CAP;
00764 }
00765
00766 void reset()
00767 {
00768 _c = L4_INVALID_CAP;
00769 }
00770
00771 void reset()
00772 {
00773 _c = L4_INVALID_CAP;
00774 }
00775
00776 void reset()
00777 {
00778 _c = L4_INVALID_CAP;
00779 }
00780
00781 void reset()
00782 {
00783 _c = L4_INVALID_CAP;
00784 }
00785
00786 void reset()
00787 {
00788 _c = L4_INVALID_CAP;
00789 }
00790
00791 void reset()
00792 {
00793 _c = L4_INVALID_CAP;
00794 }
00795
00796 void reset()
00797 {
00798 _c = L4_INVALID_CAP;
00799 }
00800
00801 void reset()
00802 {
00803 _c = L4_INVALID_CAP;
00804 }
00805
00806 void reset()
00807 {
00808 _c = L4_INVALID_CAP;
00809 }
00810
00811 void reset()
00812 {
00813 _c = L4_INVALID_CAP;
00814 }
00815
00816 void reset()
00817 {
00818 _c = L4_INVALID_CAP;
00819 }
00820
00821 void reset()
00822 {
00823 _c = L4_INVALID_CAP;
00824 }
00825
00826 void reset()
00827 {
00828 _c = L4_INVALID_CAP;
00829 }
00830
00831 void reset()
00832 {
00833 _c = L4_INVALID_CAP;
00834 }
00835
00836 void reset()
00837 {
00838 _c = L4_INVALID_CAP;
00839 }
00840
00841 void reset()
00842 {
00843 _c = L4_INVALID_CAP;
00844 }
00845
00846 void reset()
00847 {
00848 _c = L4_INVALID_CAP;
00849 }
00850
00851 void reset()
00852 {
00853 _c = L4_INVALID_CAP;
00854 }
00855
00856 void reset()
00857 {
00858 _c = L4_INVALID_CAP;
00859 }
00860
00861 void reset()
00862 {
00863 _c = L4_INVALID_CAP;
00864 }
00865
00866 void reset()
00867 {
00868 _c = L4_INVALID_CAP;
00869 }
00870
00871 void reset()
00872 {
00873 _c = L4_INVALID_CAP;
00874 }
00875
00876 void reset()
00877 {
00878 _c = L4_INVALID_CAP;
00879 }
00880
00881 void reset()
00882 {
00883 _c = L4_INVALID_CAP;
00884 }
00885
00886 void reset()
00887 {
00888 _c = L4_INVALID_CAP;
00889 }
00890
00891 void reset()
00892 {
00893 _c = L4_INVALID_CAP;
00894 }
00895
00896 void reset()
00897 {
00898 _c = L4_INVALID_CAP;
00899 }
00900
00901 void reset()
00902 {
00903 _c = L4_INVALID_CAP;
00904 }
00905
00906 void reset()
00907 {
00908 _c = L4_INVALID_CAP;
00909 }
00910
00911 void reset()
00912 {
00913 _c = L4_INVALID_CAP;
00914 }
00915
00916 void reset()
00917 {
00918 _c = L4_INVALID_CAP;
00919 }
00920
00921 void reset()
00922 {
00923 _c = L4_INVALID_CAP;
00924 }
00925
00926 void reset()
00927 {
00928 _c = L4_INVALID_CAP;
00929 }
00930
00931 void reset()
00932 {
00933 _c = L4_INVALID_CAP;
00934 }
00935
00936 void reset()
00937 {
00938 _c = L4_INVALID_CAP;
00939 }
00940
00941 void reset()
00942 {
00943 _c = L4_INVALID_CAP;
00944 }
00945
00946 void reset()
00947 {
00948 _c = L4_INVALID_CAP;
00949 }
00950
00951 void reset()
00952 {
00953 _c = L4_INVALID_CAP;
00954 }
00955
00956 void reset()
00957 {
00958 _c = L4_INVALID_CAP;
00959 }
00960
00961 void reset()
00962 {
00963 _c = L4_INVALID_CAP;
00964 }
00965
00966 void reset()
00967 {
00968 _c = L4_INVALID_CAP;
00969 }
00970
00971 void reset()
00972 {
00973 _c = L4_INVALID_CAP;
00974 }
00975
00976 void reset()
00977 {
00978 _c = L4_INVALID_CAP;
00979 }
00980
00981 void reset()
00982 {
00983 _c = L4_INVALID_CAP;
00984 }
00985
00986 void reset()
00987 {
00988 _c = L4_INVALID_CAP;
00989 }
00990
00991 void reset()
00992 {
00993 _c = L4_INVALID_CAP;
00994 }
00995
00996 void reset()
00997 {
00998 _c = L4_INVALID_CAP;
00999 }
01000
01001 void reset()
01002 {
01003 _c = L4_INVALID_CAP;
01004 }
01005
01006 void reset()
01007 {
01008 _c = L4_INVALID_CAP;
01009 }
01010
01011 void reset()
01012 {
01013 _c = L4_INVALID_CAP;
01014 }
01015
01016 void reset()
01017 {
01018 _c = L4_INVALID_CAP;
01019 }
01020
01021 void reset()
01022 {
01023 _c = L4_INVALID_CAP;
01024 }
01025
01026 void reset()
01027 {
01028 _c = L4_INVALID_CAP;
01029 }
01030
01031 void reset()
01032 {
01033 _c = L4_INVALID_CAP;
01034 }
01035
01036 void reset()
01037 {
01038 _c = L4_INVALID_CAP;
01039 }
01040
01041 void reset()
01042 {
01043 _c = L4_INVALID_CAP;
01044 }
01045
01046 void reset()
01047 {
01048 _c = L4_INVALID_CAP;
01049 }
01050
01051 void reset()
01052 {
01053 _c = L4_INVALID_CAP;
01054 }
01055
01056 void reset()
01057 {
01058 _c = L4_INVALID_CAP;
01059 }
01060
01061 void reset()
01062 {
01063 _c = L4_INVALID_CAP;
01064 }
01065
01066 void reset()
01067 {
01068 _c = L4_INVALID_CAP;
01069 }
01070
01071 void reset()
01072 {
01073 _c = L4_INVALID_CAP;
01074 }
01075
01076 void reset()
01077 {
01078 _c = L4_INVALID_CAP;
01079 }
01080
01081 void reset()
01082 {
01083 _c = L4_INVALID_CAP;
01084 }
01085
01086 void reset()
01087 {
01088 _c = L4_INVALID_CAP;
01089 }
01090
01091 void reset()
01092 {
01093 _c = L4_INVALID_CAP;
01094 }
01095
01096 void reset()
01097 {
01098 _c = L4_INVALID_CAP;
01099 }
01100
01101 void reset()
01102 {
01103 _c = L4_INVALID_CAP;
01104 }
01105
01106 void reset()
01107 {
01108 _c = L4_INVALID_CAP;
01109 }
01110
01111 void reset()
01112 {
01113 _c = L4_INVALID_CAP;
01114 }
01115
01116 void reset()
01117 {
01118 _c = L4_INVALID_CAP;
01119 }
01120
01121 void reset()
01122 {
01123 _c = L4_INVALID_CAP;
01124 }
01125
01126 void reset()
01127 {
01128 _c = L4_INVALID_CAP;
01129 }
01130
01131 void reset()
01132 {
01133 _c = L4_INVALID_CAP;
01134 }
01135
01136 void reset()
01137 {
01138 _c = L4_INVALID_CAP;
01139 }
01140
01141 void reset()
01142 {
01143 _c = L4_INVALID_CAP;
01144 }
01145
01146 void reset()
01147 {
01148 _c = L4_INVALID_CAP;
01149 }
01150
01151 void reset()
01152 {
01153 _c = L4_INVALID_CAP;
01154 }
01155
01156 void reset()
01157 {
01158 _c = L4_INVALID_CAP;
01159 }
01160
01161 void reset()
01162 {
01163 _c = L4_INVALID_CAP;
01164 }
01165
01166 void reset()
01167 {
01168 _c = L4_INVALID_CAP;
01169 }
01170
01171 void reset()
01172 {
01173 _c = L4_INVALID_CAP;
01174 }
01175
01176 void reset()
01177 {
01178 _c = L4_INVALID_CAP;
01179 }
01180
01181 void reset()
01182 {
01183 _c = L4_INVALID_CAP;
01184 }
01185
01186 void reset()
01187 {
01188 _c = L4_INVALID_CAP;
01189 }
01190
01191 void reset()
01192 {
01193 _c = L4_INVALID_CAP;
01194 }
01195
01196 void reset()
01197 {
01198 _c = L4_INVALID_CAP;
01199 }
01200
01201 void reset()
01202 {
01203 _c = L4_INVALID_CAP;
01204 }
01205
01206 void reset()
01207 {
01208 _c = L4_INVALID_CAP;
01209 }
01210
01211 void reset()
01212 {
01213 _c = L4_INVALID_CAP;
01214 }
01215
01216 void reset()
01217 {
01218 _c = L4_INVALID_CAP;
01219 }
01220
01221 void reset()
01222 {
01223 _c = L4_INVALID_CAP;
01224 }
01225
01226 void reset()
01227 {
01228 _c = L4_INVALID_CAP;
01229 }
01230
01231 void reset()
01232 {
01233 _c = L4_INVALID_CAP
```



```

00058 }
00059
00060 Smart_cap() throw() : Cap_base(Invalid) {}
00061
00062 Smart_cap(Cap_base::Cap_type t) throw() : Cap_base(t) {}
00063
00072 template< typename O >
00073 Smart_cap(Cap<O> const &p) throw() : Cap_base(p.cap())
00074 { T* __t = ((O*)100); (void)__t; }
00075
00076 template< typename O >
00077 Smart_cap(Cap<O> const &p, SMART const &smart) throw()
00078 : Cap_base(p.cap()), SMART(smart)
00079 { T* __t = ((O*)100); (void)__t; }
00080
00081 template< typename O >
00082 Smart_cap(Smart_cap<O, SMART> const &o) throw()
00083 : Cap_base(SMART::copy(o), SMART(o.smart()))
00084 { T* __t = ((O*)100); (void)__t; }
00085
00086 Smart_cap(Smart_cap const &o) throw()
00087 : Cap_base(SMART::copy(o), SMART(o.smart()))
00088 { }
00089
00090 template< typename O >
00091 Smart_cap(typename Cap<O>::Cap_type cap) throw() :
00092 Cap_base(cap)
00093 { T* __t = ((O*)100); (void)__t; }
00094
00095 void operator = (typename Cap<T>::Cap_type cap) throw()
00096 {
00097 _delete();
00098 _c = cap;
00099 }
00100
00101 template< typename O >
00102 void operator = (Smart_cap<O, SMART> const &o) throw()
00103 {
00104 _delete();
00105 _c = this->SMART::copy(o).cap();
00106 this->SMART::operator = (o.smart());
00107 // return *this;
00108 }
00109
00110 Smart_cap const &operator = (Smart_cap const &o) throw()
00111 {
00112 if (&o == this)
00113 return *this;
00114
00115 _delete();
00116 _c = this->SMART::copy(o).cap();
00117 this->SMART::operator = (o.smart());
00118 return *this;
00119 }
00120
00121 #if __cplusplus >= 201103L
00122 template< typename O >
00123 Smart_cap(Smart_cap<O, SMART> &&o) throw()
00124 : Cap_base(o.release()), SMART(o.smart())
00125 { T* __t = ((O*)100); (void)__t; }
00126
00127 Smart_cap(Smart_cap &&o) throw()
00128 : Cap_base(o.release()), SMART(o.smart())
00129 { }
00130
00131 template< typename O >
00132 void operator = (Smart_cap<O, SMART> &&o) throw()
00133 {
00134 _delete();
00135 _c = o.release().cap();
00136 this->SMART::operator = (o.smart());
00137 // return *this;
00138 }
00139
00140 Smart_cap const &operator = (Smart_cap &&o) throw()
00141 {
00142 if (&o == this)
00143 return *this;
00144
00145 _delete();
00146 _c = o.release().cap();
00147 this->SMART::operator = (o.smart());
00148 return *this;
00149 }
00150 #endif
00151
00152 Cap<T> operator -> () const throw() { return Cap<T>(_c); }

```

```

00155
00156 Cap<T> get() const throw() { return Cap<T>(_c); }
00157
00158 ~Smart_cap() throw() { _delete(); }
00159 };
00160
00161 template< typename T >
00162 class Weak_cap : public Cap_base
00163 {
00164 public:
00165 Weak_cap() : Cap_base(Invalid) {}
00166
00167 template< typename O >
00168 Weak_cap(typename Cap<O>::Cap_type t) : Cap_base(t)
00169 { T* __t = ((O*)100); (void)__t; }
00170
00171 template< typename O, typename S >
00172 Weak_cap(Smart_cap<O, S> const &c) : Cap_base(c.cap())
00173 { T* __t = ((O*)100); (void)__t; }
00174
00175 Weak_cap(Weak_cap const &o) : Cap_base(o) {}
00176
00177 template< typename O >
00178 Weak_cap(Weak_cap<O> const &o) : Cap_base(o)
00179 { T* __t = ((O*)100); (void)__t; }
00180
00181 };
00182
00183 namespace Cap_traits {
00184 template< typename T1, typename T2 >
00185 struct Type { enum { Equal = false }; };
00186
00187 template< typename T1 >
00188 struct Type<T1,T1> { enum { Equal = true }; };
00189 };
00190
00201 template< typename T, typename F, typename SMART >
00202 inline
00203 Smart_cap<T, SMART> cap_cast(Smart_cap<F, SMART> const &c)
00204 throw()
00205 {
00206 (void)static_cast<T const *>(reinterpret_cast<F const *>(100));
00207 return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00208 }
00209
00220 template< typename T, typename F, typename SMART >
00221 inline
00222 Smart_cap<T, SMART> cap_reinterpret_cast(
00223 Smart_cap<F, SMART> const &c) throw()
00224 {
00225 return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00226 }
00227
00228 }
00229

```

## 15.359 l4/sys/task File Reference

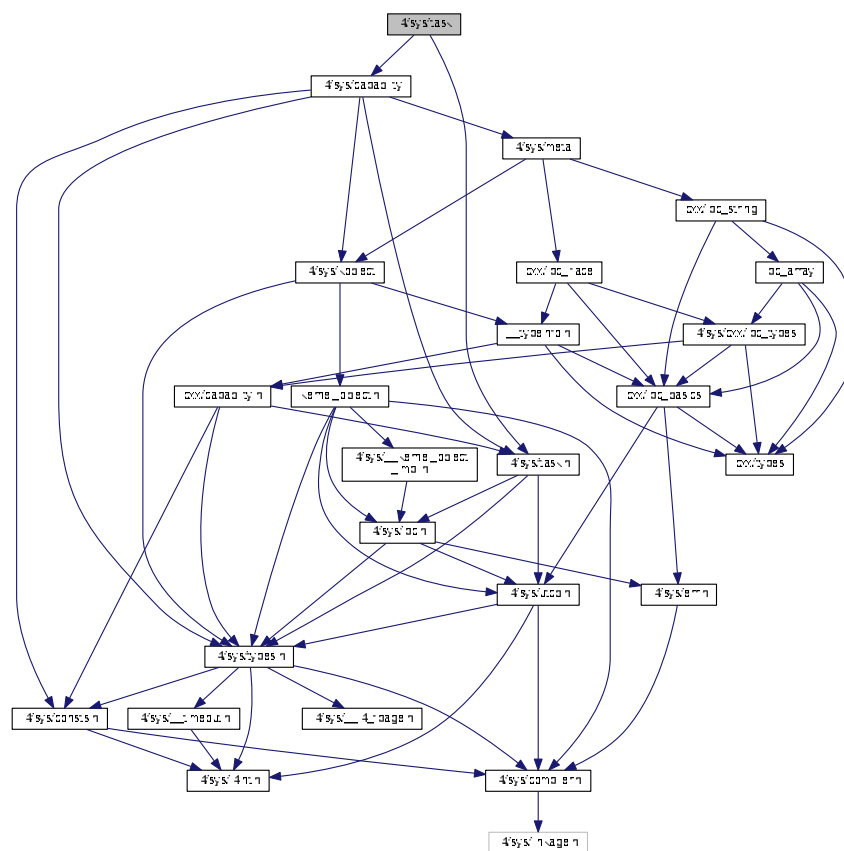
Common task related definitions.

```

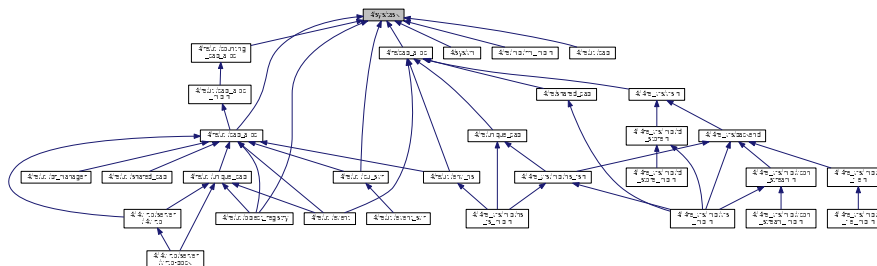
#include <l4/sys/task.h>
#include <l4/sys/capability>

```

Include dependency graph for task:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Task`  
*C++ interface of the `Task` kernel object.*

## Namespaces

- L4  
*L4 low-level kernel interface.*

## 15.359.1 Detailed Description

Common task related definitions.

Definition in file [task](#).

## 15.360 task

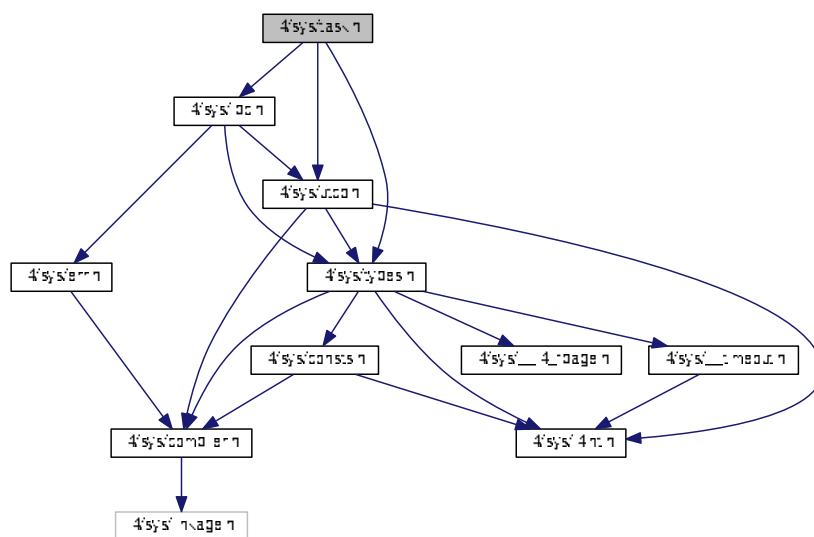
```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task.h>
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00043 class Task :
00044 public Kobject<Task, Kobject, L4_PROTO_TASK,
00045 Type_info::Demand_t<2> >
00046 {
00047 public:
00067 l4_msgtag_t map(Cap<Task> const &src_task,
00068 l4_fpage_t const &snd_fpage, l4_addr_t snd_base,
00069 l4_utcb_t *utcb = l4_utcb()) throw()
00070 { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00071
00090 l4_msgtag_t unmap(l4_fpage_t const &fpage,
00091 l4_umword_t map_mask,
00092 l4_utcb_t *utcb = l4_utcb()) throw()
00093 { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00094
00115 l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00116 unsigned num_fpages,
00117 l4_umword_t map_mask,
00118 l4_utcb_t *utcb = l4_utcb()) throw()
00119 { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00120
00133 l4_msgtag_t delete_obj(L4::Cap<void> obj,
00134 l4_utcb_t *utcb = l4_utcb()) throw()
00135 { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00136
00147 l4_msgtag_t release_cap(L4::Cap<void> cap,
00148 l4_utcb_t *utcb = l4_utcb()) throw()
00149 { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00150
00167 l4_msgtag_t cap_valid(Cap<void> const &cap,
00168 l4_utcb_t *utcb = l4_utcb()) throw()
00169 { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00170
00183 l4_msgtag_t cap_has_child(Cap<void> const &cap,
00184 l4_utcb_t *utcb = l4_utcb()) throw()
00185 { L4_DEPRECATED("Do not use. Future uncertain.");
00186 return l4_task_cap_has_child_u(this->cap(), cap.cap(), utcb); }
00187
00200 l4_msgtag_t cap_equal(Cap<void> const &cap_a,
00201 Cap<void> const &cap_b,
00202 l4_utcb_t *utcb = l4_utcb()) throw()
00203 { return l4_task_cap_equal_u(cap(), cap_a.cap(), cap_b.cap(), utcb); }
00204
00219 l4_msgtag_t add_ku_mem(l4_fpage_t const &fpage,

```

## 15.361 I4/sys/task.h File Reference

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
Include dependency graph for task.h:
```



- enum `L4_task_ops` {  
`L4_TASK_MAP_OP` = 0UL, `L4_TASK_UNMAP_OP` = 1UL, `L4_TASK_CAP_INFO_OP` = 2UL, `L4_TASK_ADD_KU_MEM_OP` = 3UL,  
`L4_TASK_LDT_SET_X86_OP` = 0x11UL }  
*Operations on task objects.*

## Functions

- `l4_msgtag_t l4_task_map (l4_cap_idx_t dst_task, l4_cap_idx_t src_task, l4_fpage_t snd_fpage, l4_addr_t snd_base) L4_NOTHROW`  
*Map resources available in the source task to a destination task.*
- `l4_msgtag_t l4_task_unmap (l4_cap_idx_t task, l4_fpage_t fpage, l4_umword_t map_mask) L4_NOTHROW`  
*Revoke rights from the task.*
- `l4_msgtag_t l4_task_unmap_batch (l4_cap_idx_t task, l4_fpage_t const *fpages, unsigned num_fpages, unsigned long map_mask) L4_NOTHROW`  
*Revoke rights from a task.*
- `l4_msgtag_t l4_task_delete_obj (l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW`  
*Release capability and delete object.*
- `l4_msgtag_t l4_task_release_cap (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`  
*Release capability.*
- `l4_msgtag_t l4_task_cap_valid (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`  
*Check whether a capability is present (refers to an object).*
- `l4_msgtag_t l4_task_cap_has_child (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`  
*Test whether a capability has child mappings (in another task).*
- `l4_msgtag_t l4_task_cap_equal (l4_cap_idx_t task, l4_cap_idx_t cap_a, l4_cap_idx_t cap_b) L4_NOTHROW`  
*Test whether two capabilities point to the same object with the same rights.*
- `l4_msgtag_t l4_task_add_ku_mem (l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW`  
*Add kernel-user memory.*

### 15.361.1 Detailed Description

Common task related definitions.

Definition in file [task.h](#).

## 15.362 task.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/utcb.h>
00028
00060 L4_INLINE l4_msgtag_t
00061 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00062 l4_fpage_t snd_fpage, l4_addr_t snd_base)
00063 L4_NOTHROW;
```

```

00063
00067 L4_INLINE l4_msgtag_t
00068 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00069 l4_fpage_t snd_fpage, l4_addr_t snd_base,
00070 l4_utcb_t *utcb) L4_NOTHROW;
00070
00089 L4_INLINE l4_msgtag_t
00090 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00091 l4_umword_t map_mask) L4_NOTHROW;
00092
00096 L4_INLINE l4_msgtag_t
00097 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00098 l4_umword_t map_mask, l4_utcb_t *utcb)
00099 L4_NOTHROW;
00099
00123 L4_INLINE l4_msgtag_t
00124 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00125 unsigned num_fpages, unsigned long map_mask) L4_NOTHROW;
00126
00130 L4_INLINE l4_msgtag_t
00131 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00132 unsigned num_fpages, unsigned long map_mask,
00133 l4_utcb_t *u) L4_NOTHROW;
00134
00150 L4_INLINE l4_msgtag_t
00151 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj)
00152 L4_NOTHROW;
00152
00156 L4_INLINE l4_msgtag_t
00157 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00158 l4_utcb_t *u) L4_NOTHROW;
00159
00171 L4_INLINE l4_msgtag_t
00172 l4_task_release_cap(l4_cap_idx_t task,
00173 l4_cap_idx_t cap) L4_NOTHROW;
00173
00177 L4_INLINE l4_msgtag_t
00178 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00179 l4_utcb_t *u) L4_NOTHROW;
00180
00181
00198 L4_INLINE l4_msgtag_t
00199 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap)
00200 L4_NOTHROW;
00200
00204 L4_INLINE l4_msgtag_t
00205 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00206 l4_utcb_t *utcb) L4_NOTHROW;
00206
00219 L4_INLINE l4_msgtag_t
00220 l4_task_cap_has_child(l4_cap_idx_t task,
00221 l4_cap_idx_t cap) L4_NOTHROW
00222 L4_DEPRECATED("Do not use. Future uncertain.");
00222
00226 L4_INLINE l4_msgtag_t
00227 l4_task_cap_has_child_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00228 l4_utcb_t *utcb) L4_NOTHROW;
00228
00241 L4_INLINE l4_msgtag_t
00242 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00243 l4_cap_idx_t cap_b) L4_NOTHROW;
00244
00248 L4_INLINE l4_msgtag_t
00249 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
00250 l4_utcb_t *u) L4_NOTHROW;
00251
00261 L4_INLINE l4_msgtag_t
00262 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem)
00263 L4_NOTHROW;
00263
00264
00268 L4_INLINE l4_msgtag_t
00269 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00270 l4_cap_idx_t cap_b, l4_utcb_t *utcb)
00271 L4_NOTHROW;
00271
00276 enum l4_task_ops
00277 {
00278 L4_TASK_MAP_OP = 0UL,
00279 L4_TASK_UNMAP_OP = 1UL,
00280 L4_TASK_CAP_INFO_OP = 2UL,
00281 L4_TASK_ADD_KU_MEM_OP = 3UL,
00282 L4_TASK_LDT_SET_X86_OP = 0x11UL,
00283 };
00284
00285
00286 /* IMPLEMENTATION ----- */

```

```

00287
00288 #include <l4/sys/ipc.h>
00289
00290
00291 L4_INLINE l4_msgtag_t
00292 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00293 l4_fpage_t snd_fpage, unsigned long snd_base, l4_utcb_t *u)
00294 L4_NOTHROW
00295 {
00296 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00297 v->mr[0] = L4_TASK_MAP_OP;
00298 v->mr[3] = l4_map_obj_control(0,0);
00299 v->mr[4] = l4_obj_fpage(src_task, 0, L4_FPAGE_RWX).
00300 raw;
00301 v->mr[1] = snd_base;
00302 v->mr[2] = snd_fpage.raw;
00303 return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0),
00304 L4_IPC_NEVER);
00305 }
00306
00307 L4_INLINE l4_msgtag_t
00308 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00309 unsigned long map_mask, l4_utcb_t *u) L4_NOTHROW
00310 {
00311 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00312 v->mr[0] = L4_TASK_UNMAP_OP;
00313 v->mr[1] = map_mask;
00314 v->mr[2] = fpage.raw;
00315 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0),
00316 L4_IPC_NEVER);
00317 }
00318
00319 L4_INLINE l4_msgtag_t
00320 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00321 unsigned num_fpages, unsigned long map_mask,
00322 l4_utcb_t *u) L4_NOTHROW
00323 {
00324 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00325 v->mr[0] = L4_TASK_UNMAP_OP;
00326 v->mr[1] = map_mask;
00327 __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
00328 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0,
00329 0), L4_IPC_NEVER);
00330 }
00331
00332 L4_INLINE l4_msgtag_t
00333 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00334 l4_utcb_t *u) L4_NOTHROW
00335 {
00336 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00337 v->mr[0] = L4_TASK_CAP_INFO_OP;
00338 v->mr[1] = cap & ~1UL;
00339 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0),
00340 L4_IPC_NEVER);
00341 }
00342
00343 L4_INLINE l4_msgtag_t
00344 l4_task_cap_has_child_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00345 l4_utcb_t *u) L4_NOTHROW
00346 {
00347 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00348 v->mr[0] = L4_TASK_CAP_INFO_OP;
00349 v->mr[1] = cap | 1UL;
00350 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0),
00351 L4_IPC_NEVER);
00352 }
00353
00354 L4_INLINE l4_msgtag_t
00355 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00356 l4_cap_idx_t cap_b, l4_utcb_t *u)
00357 L4_NOTHROW
00358 {
00359 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00360 v->mr[0] = L4_TASK_CAP_INFO_OP;
00361 v->mr[1] = cap_a;
00362 v->mr[2] = cap_b;
00363 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0),
00364 L4_IPC_NEVER);
00365 }
00366
00367 L4_INLINE l4_msgtag_t
00368 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
00369 l4_utcb_t *u) L4_NOTHROW
00370 {
00371 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00372 v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
00373 v->mr[1] = ku_mem.raw;

```



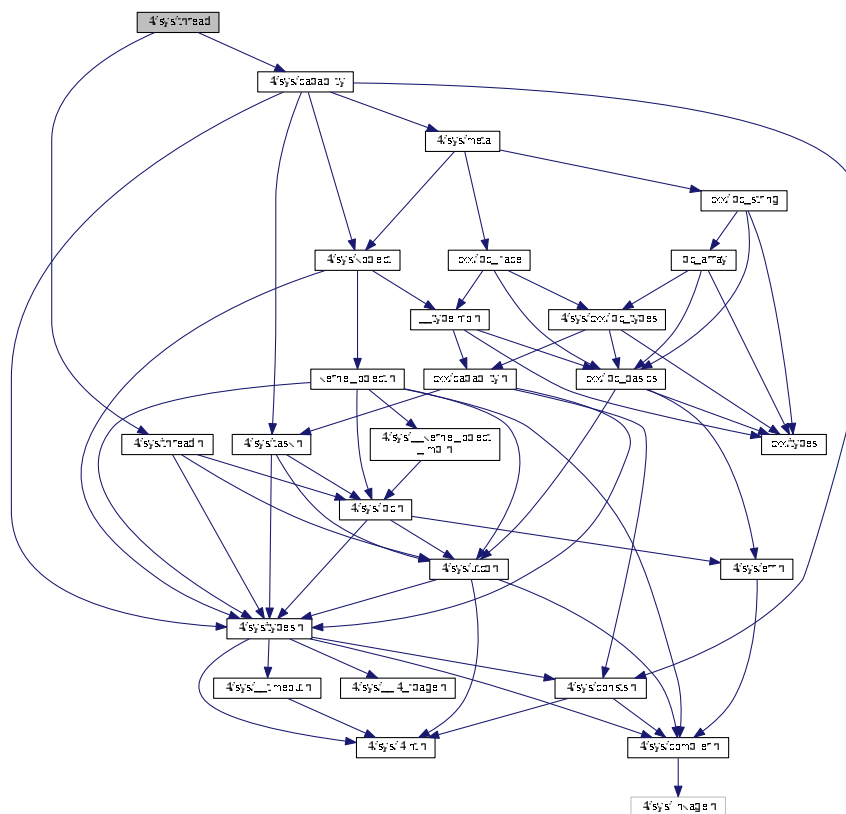
```

00363 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0),
00364 L4_IPC_NEVER);
00365 }
00366
00367
00368 L4_INLINE l4_msgtag_t
00369 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00370 l4_fpage_t snd_fpage, unsigned long snd_base) L4_NOTHROW
00371 {
00372 return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
00373 }
00374
00375 L4_INLINE l4_msgtag_t
00376 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00377 unsigned long map_mask) L4_NOTHROW
00378 {
00379 return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
00380 }
00381
00382 L4_INLINE l4_msgtag_t
00383 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00384 unsigned num_fpages, unsigned long map_mask) L4_NOTHROW
00385 {
00386 return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
00387 l4_utcb());
00388 }
00389
00390 L4_INLINE l4_msgtag_t
00391 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00392 l4_utcb_t *u) L4_NOTHROW
00393 {
00394 return l4_task_unmap_u(task, l4_obj_fpage(obj, 0,
00395 L4_CAP_FPAGE_RWSD),
00396 L4_FP_DELETE_OBJ, u);
00397 }
00398
00399 L4_INLINE l4_msgtag_t
00400 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj)
00401 L4_NOTHROW
00402 {
00403 return l4_task_delete_obj_u(task, obj, l4_utcb());
00404 }
00405
00406 L4_INLINE l4_msgtag_t
00407 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00408 l4_utcb_t *u) L4_NOTHROW
00409 {
00410 return l4_task_unmap_u(task, l4_obj_fpage(cap, 0,
00411 L4_CAP_FPAGE_RWSD),
00412 L4_FP_ALL_SPACES, u);
00413 }
00414
00415 L4_INLINE l4_msgtag_t
00416 l4_task_release_cap(l4_cap_idx_t task,
00417 l4_cap_idx_t cap) L4_NOTHROW
00418 {
00419 return l4_task_release_cap_u(task, cap, l4_utcb());
00420 }
00421
00422 L4_INLINE l4_msgtag_t
00423 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap)
00424 L4_NOTHROW
00425 {
00426 return l4_task_cap_valid_u(task, cap, l4_utcb());
00427 }
00428
00429 L4_INLINE l4_msgtag_t
00430 l4_task_cap_has_child(l4_cap_idx_t task,
00431 l4_cap_idx_t cap) L4_NOTHROW
00432 {
00433 return l4_task_cap_has_child_u(task, cap, l4_utcb());
00434 }
00435
00436 L4_INLINE l4_msgtag_t
00437 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00438 l4_cap_idx_t cap_b) L4_NOTHROW
00439 {
00440 return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00441 }
00442
00443 L4_INLINE l4_msgtag_t
00444 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem)
00445 L4_NOTHROW
00446 {
00447 return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00448 }

```

### 15.363 I4/sys/thread File Reference

```
#include <linux/sys/capability>
#include <linux/sys/thread.h>
Include dependency graph for thread:
```



```
405: sysfs_read
 ^
 |
 |__ 404: read_from_reg_sysfs
```

## Data Structures

- class [L4::Thread](#)  
*C++ L4 kernel thread interface.*
- class [L4::Thread::Attr](#)  
*Thread attributes used for control\_commit().*
- class [L4::Thread::Modify\\_senders](#)  
*Wrapper class for modifying senders.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 15.363.1 Detailed Description

Common thread related definitions.

Definition in file [thread](#).

## 15.364 thread

```

00001 // vi:set ft=c++: -- Mode: C++ --
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/sys/thread.h>
00029
00030 namespace L4 {
00031
00058 class Thread :
00059 public Kobject<Thread, Kobject, L4_PROTO_THREAD,
00060 Type_info::Demand_t<1> >
00061 {
00062 public:
00085 l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00086 l4_umword_t flags,
00087 l4_utcb_t *utcb = l4_utcb()) throw()
00088 { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00089
00111 l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00112 l4_umword_t *flags,
00113 l4_utcb_t *utcb = l4_utcb()) throw()
00114 { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00115
00116
00125 class Attr

```

```

00126 {
00127 private:
00128 friend class L4::Thread;
00129 l4_utcb_t *_u;
00130
00131 public:
00132 explicit Attr(l4_utcb_t *utcb = l4_utcb()) throw() : _u(utcb)
00133 { l4_thread_control_start_u(utcb); }
00134
00135 void pager(Cap<void> const &pager) throw()
00136 { l4_thread_control_pager_u(pager.cap(), _u); }
00137
00138 Cap<void> pager() throw()
00139 { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }
00140
00141 void exc_handler(Cap<void> const &exc_handler) throw()
00142 { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }
00143
00144 Cap<void> exc_handler() throw()
00145 { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }
00146
00147 void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) throw()
00148 { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }
00149
00150 void alien(int on) throw()
00151 { l4_thread_control_alien_u(_u, on); }
00152
00153 void ux_host_syscall(int on) throw()
00154 { l4_thread_control_ux_host_syscall_u(_u, on); }
00155
00156 };
00157
00158 l4_msgtag_t control(Attr const &attr) throw()
00159 { return l4_thread_control_commit_u(cap(), attr._u); }
00160
00161 l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) throw()
00162 { return l4_thread_switch_u(cap(), utcb); }
00163
00164 l4_msgtag_t stats_time(l4_kernel_clock_t *us,
00165 l4_utcb_t *utcb = l4_utcb()) throw()
00166 { return l4_thread_stats_time_u(cap(), us, utcb); }
00167
00168 l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb =
00169 l4_utcb()) throw()
00170 { return l4_thread_vcpu_resume_start_u(utcb); }
00171
00172 l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
00173 l4_utcb_t *utcb = l4_utcb()) throw()
00174 { return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }
00175
00176 l4_msgtag_t vcpu_control(l4_addr_t vcpu_state,
00177 l4_utcb_t *utcb = l4_utcb())
00178 throw()
00179 { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }
00180
00181 l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
00182 l4_utcb_t *utcb = l4_utcb()) throw()
00183 { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }
00184
00185 l4_msgtag_t register_del_irq(Cap<Irq> irq,
00186 l4_utcb_t *u = l4_utcb()) throw()
00187 { return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }
00188
00189 class Modify_senders
00190 {
00191 private:
00192 friend class Thread;
00193 l4_utcb_t *utcb;
00194 unsigned cnt;
00195
00196 public:
00197 explicit Modify_senders(l4_utcb_t *u = l4_utcb()) throw()
00198 : utcb(u), cnt(1)
00199 {
00200 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00201 }
00202
00203 int add(l4_umword_t match_mask, l4_umword_t match,
00204 l4_umword_t del_bits, l4_umword_t add_bits) throw()
00205 {
00206 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00207 if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00208 return -L4_ENOMEM;
00209 m->mr[cnt++] = match_mask;
00210 m->mr[cnt++] = match;
00211 m->mr[cnt++] = del_bits;
00212 m->mr[cnt++] = add_bits;
00213 }
00214

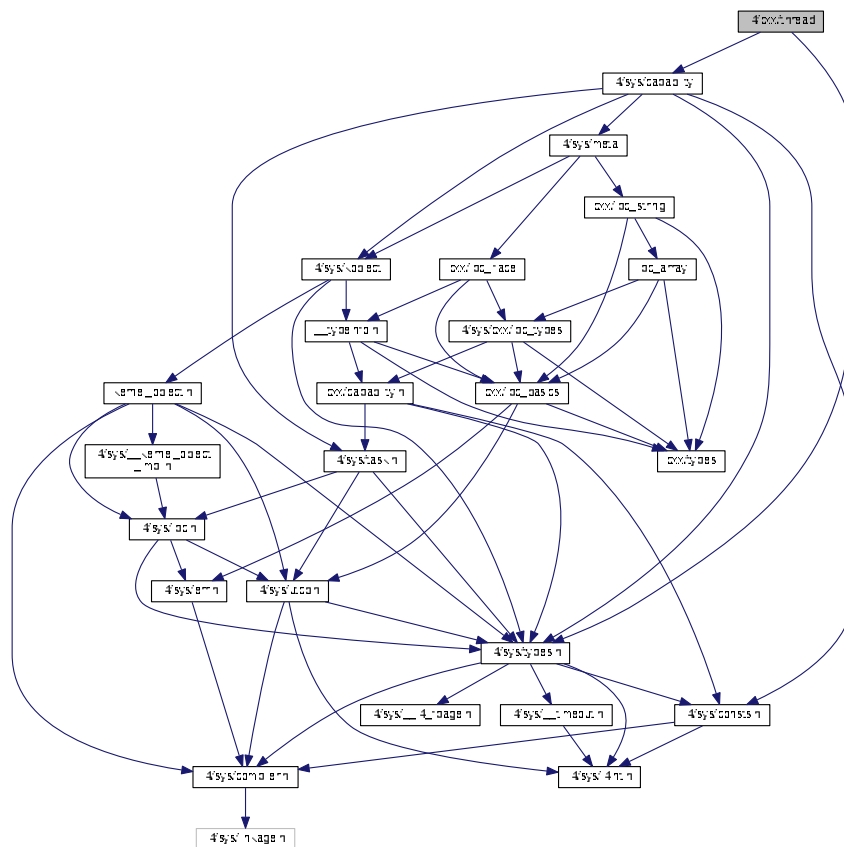
```

```
00362 return 0;
00363 }
00364 };
00365
00373 l4_msgtag_t modify_senders(Modify_senders const &todo) throw()
00374 {
00375 return l4_ipc_call(cap(), todo.utcb, l4_msgtag(
00376 L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
00377 };
00378 }
```

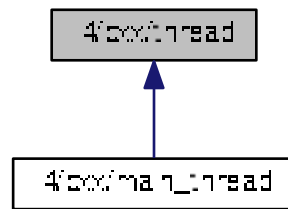
### 15.365 I4/cxx/thread File Reference

### Thread implementation.

```
#include <linux/sys/capability>
#include <linux/sys/types.h>
Include dependency graph for thread:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [cxx](#)

*Our C++ library.*

### 15.365.1 Detailed Description

Thread implementation.

Definition in file [thread](#).

## 15.366 thread

```

00001
00005 /*
00006 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012
00013 #ifndef CXX_THREAD_H__
00014 #define CXX_THREAD_H__
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/types.h>
00018
00019 namespace cxx {
00020
00021 class Thread
00022 {
00023 public:
00024
00025 enum State
00026 {
00027 Dead = 0,
00028 Running = 1,
00029 Stopped = 2,
00030 };
00031
00032 Thread(bool initiate);
00033 Thread(void *stack);
00034 Thread(void *stack, L4::Cap<L4::Thread> const &cap);
00035 virtual ~Thread();
00036 void execute() asm ("L4_Thread_execute");

```

```

00037 virtual void run() = 0;
00038 virtual void shutdown() asm ("L4_Thread_shutdown");
00039 void start();
00040 void stop();
00041
00042 L4::Cap<L4::Thread> self() const throw()
00043 { return _cap; }
00044
00045 State state() const
00046 { return _state; }
00047
00048 static void start_cxx_thread(Thread *_this)
00049 asm ("L4_Thread_start_cxx_thread");
00050
00051 static void kill_cxx_thread(Thread *_this)
00052 asm ("L4_Thread_kill_cxx_thread");
00053
00054 static void set_pager(L4::Cap<void>const &p) throw()
00055 { _pager = p; }
00056
00057 private:
00058 int create();
00059
00060 L4::Cap<L4::Thread> _cap;
00061 State _state;
00062
00063 protected:
00064 void *_stack;
00065
00066 private:
00067 static L4::Cap<void> _pager;
00068 static L4::Cap<void> _master;
00069 };
00070
00071 };
00072
00073 #endif /* CXX_THREAD_H__ */
00074

```

## 15.367 l4/sys/typeinfo\_svr File Reference

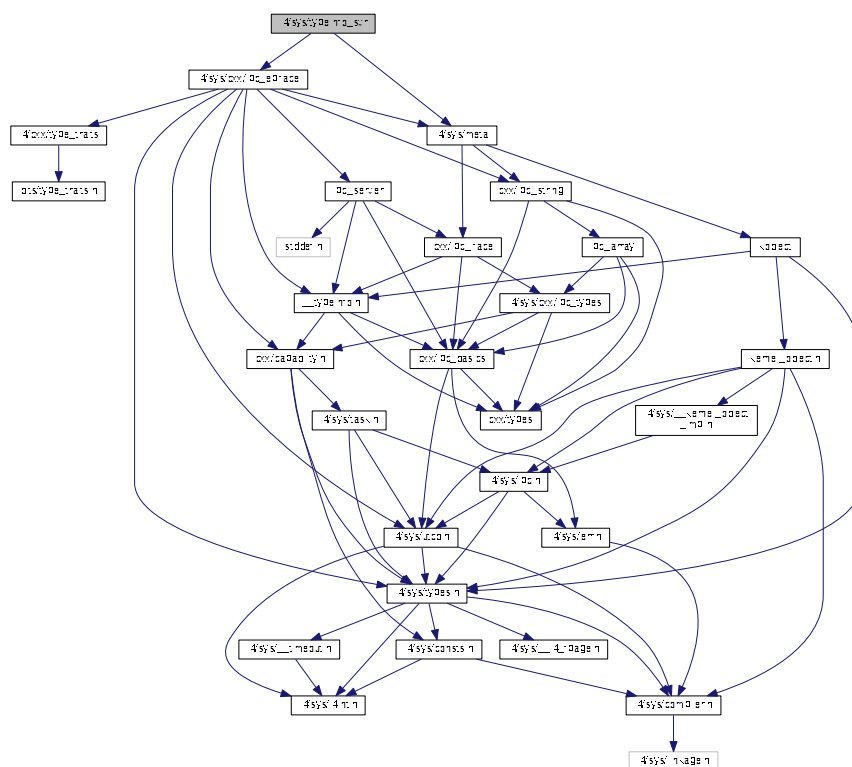
Type information server template.

```

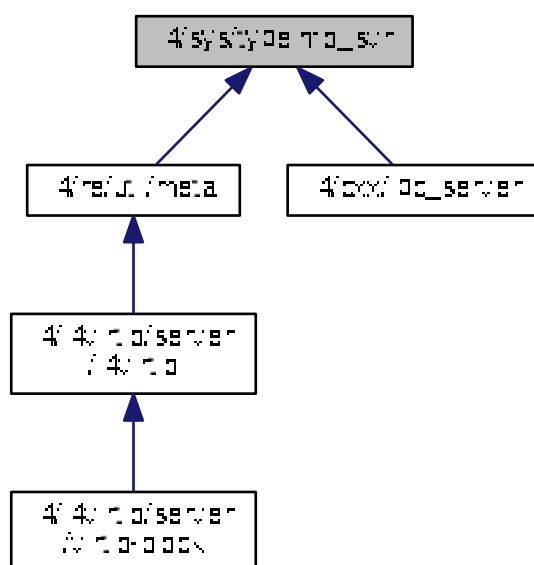
#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>

```

Include dependency graph for typeinfo\_svr:



This graph shows which files directly or indirectly include this file:





## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

### 15.367.1 Detailed Description

Type information server template.

Definition in file [typeinfo\\_svr](#).

## 15.368 typeinfo\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023
00024 #pragma once
00025
00026 #include <l4/sys/meta>
00027 #include <l4/sys/cxx/ipc_epiface>
00028
00029 namespace L4 { namespace Util {
00030
00031 template<typename KO, typename IOS>
00032 long handle_meta_request(IOS &ios)
00033 {
00034 using L4::Ipc::Msg::dispatch_call;
00035 typedef L4::Ipc::Detail::Meta_svr<KO> Msvr;
00036 l4_msgtag_t tag = dispatch_call<L4::Meta::Rpcs>((Msvr *)0, ios.utcb(),
00037 ios.tag(), 0);
00038 ios.set_ipc_params(tag);
00039 return tag.label();
00040 }
00041
00042 }}
```

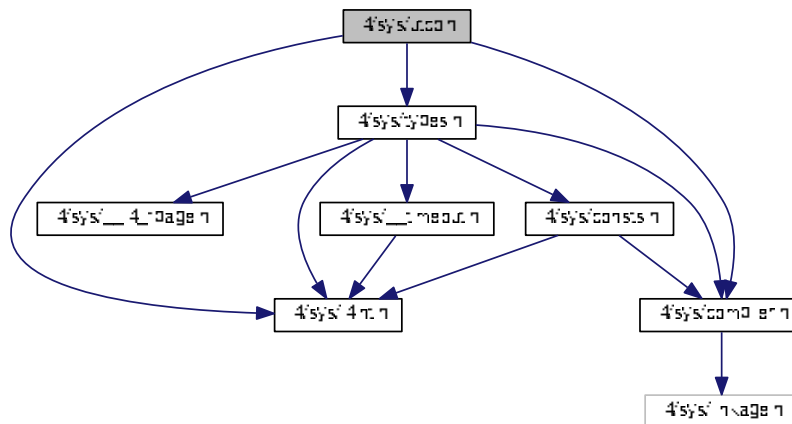
## 15.369 l4/sys/utcb.h File Reference

UTCB definitions.

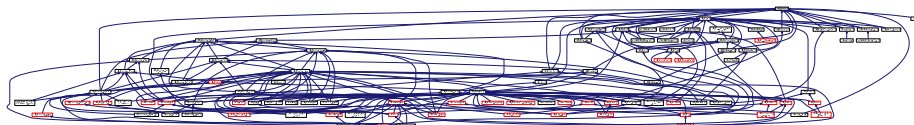
```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
Include dependency graph for utcb.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- union [l4\\_msg\\_regs\\_t](#)  
*Encapsulation of the message-register block in the UTCB.*
- struct [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*
- struct [l4\\_thread\\_regs\\_t](#)  
*Encapsulation of the thread-control-register block of the UTCB.*

## Typedefs

- typedef struct [l4\\_utcb\\_t](#) [l4\\_utcb\\_t](#)  
*Opaque type for the UTCB.*
- typedef union [l4\\_msg\\_regs\\_t](#) [l4\\_msg\\_regs\\_t](#)  
*Encapsulation of the message-register block in the UTCB.*
- typedef struct [l4\\_buf\\_regs\\_t](#) [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*
- typedef struct [l4\\_thread\\_regs\\_t](#) [l4\\_thread\\_regs\\_t](#)  
*Encapsulation of the thread-control-register block of the UTCB.*

## Functions

- `l4_utcb_t * l4_utcb (void) L4_NOTHROW L4_PURE`  
*Get the UTCB address.*
- `l4_msg_regs_t * l4_utcb_mr (void) L4_NOTHROW L4_PURE`  
*Get the message-register block of a UTCB.*
- `l4_buf_regs_t * l4_utcb_br (void) L4_NOTHROW L4_PURE`  
*Get the buffer-register block of a UTCB.*
- `l4_thread_regs_t * l4_utcb_tcr (void) L4_NOTHROW L4_PURE`  
*Get the thread-control-register block of a UTCB.*
- `l4_exc_regs_t * l4_utcb_exc (void) L4_NOTHROW L4_PURE`  
*Get the message-register block of a UTCB (for an exception IPC).*
- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
*Access function to get the program counter of the exception state.*
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`  
*Set the program counter register in the exception state.*
- `unsigned long l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
*Get the value out of an exception UTCB that describes the type of exception.*
- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
*Check whether an exception IPC is a page fault.*
- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
*Function to get the L4 style page fault address out of an exception.*
- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
*Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.*
- `void l4_utcb_inherit_fpu (int switch_on) L4_NOTHROW`  
*Enable or disable inheritance of FPU state to receiver.*
- `l4_timeout_s l4_timeout_abs (l4_kernel_clock_t pint, int br) L4_NOTHROW`  
*Set an absolute timeout.*
- `unsigned l4_utcb_mr64_idx (unsigned idx) L4_NOTHROW`  
*Get index into 64bit message registers alias from native-sized index.*

### 15.369.1 Detailed Description

UTCB definitions.

Definition in file [utcb.h](#).

## 15.370 utcb.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by

```

```

00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 /*****
00027 #ifndef _L4_SYS_UTCB_H
00028 #define _L4_SYS_UTCB_H
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/compiler.h>
00032 #include <l4/sys/l4int.h>
00033
00067 typedef struct l4_utcb_t l4_utcb_t;
00068
00078 typedef union l4_msg_regs_t
00079 {
00080 l4_umword_t mr[L4_UTCB_GENERIC_DATA_SIZE];
00081 l4_uint64_t mr64[L4_UTCB_GENERIC_DATA_SIZE / (sizeof(
00082 l4_uint64_t)/sizeof(l4_umword_t))];
00083 } l4_msg_regs_t;
00083
00093 typedef struct l4_buf_regs_t
00094 {
00096 l4_umword_t bdr;
00097
00099 l4_umword_t br[L4_UTCB_GENERIC_BUFFERS_SIZE];
00100 } l4_buf_regs_t;
00101
00110 typedef struct l4_thread_regs_t
00111 {
00113 l4_umword_t error;
00115 l4_timeout_t xfer;
00117 l4_umword_t user[3];
00118 } l4_thread_regs_t;
00119
00120 __BEGIN_DECLS
00121
00132 L4_CV l4_utcb_t *l4_utcb_wrap(void) L4_NOTHROW L4_PURE;
00133
00139 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW L4_PURE;
00140
00145 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW L4_PURE;
00146
00152 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW L4_PURE;
00153
00158 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00159
00166 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW L4_PURE;
00167
00172 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u)
00173 L4_NOTHROW L4_PURE;
00173
00179 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void)
00180 L4_NOTHROW L4_PURE;
00180
00185 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u)
00186 L4_NOTHROW L4_PURE;
00186
00199 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW L4_PURE;
00200
00205 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u)
00206 L4_NOTHROW L4_PURE;
00206
00214 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00215 L4_NOTHROW L4_PURE;
00215
00224 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00225 l4_addr_t pc) L4_NOTHROW;
00225
00230 L4_INLINE unsigned long l4_utcb_exc_typeval(l4_exc_regs_t const *u)
00231 L4_NOTHROW L4_PURE;
00231
00241 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00242
00247 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00248 L4_PURE;
00248
00249
00260 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00261 l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00261
00266 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW;
00267
00271 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on)
00272 L4_NOTHROW;
00272
00287 L4_INLINE

```

```

00288 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t pint, int br,
00289 l4_utcb_t *utcb) L4_NOTHROW;
00303 L4_INLINE
00304 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t pint, int br)
00305 L4_NOTHROW;
00313 L4_INLINE
00314 unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW;
00315
00316 /*****
00317 * Implementations
00318 *****/
00319
00320 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW
00321 { return (l4_msg_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00322
00323 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW
00324 { return (l4_buf_regs_t*)((char*)u + L4_UTCB_BUF_REGS_OFFSET); }
00325
00326 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW
00327 { return (l4_thread_regs_t*)((char*)u +
00328 L4_UTCB_THREAD_REGS_OFFSET); }
00329
00329 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW
00330 { return (l4_exc_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00331
00332 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW
00333 {
00334 if (switch_on)
00335 l4_utcb_br_u(u)->bdr |= L4_UTCB_INHERIT_FPU;
00336 else
00337 l4_utcb_br_u(u)->bdr &= ~L4_UTCB_INHERIT_FPU;
00338 }
00339
00340 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW
00341 {
00342 #ifdef L4SYS_USE_UTCB_WRAP
00343 return l4_utcb_wrap();
00344 #else
00345 return l4_utcb_direct();
00346 #endif
00347 }
00348
00349
00350
00351
00352 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW
00353 { return l4_utcb_mr_u(l4_utcb()); }
00354
00355 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW
00356 { return l4_utcb_br_u(l4_utcb()); }
00357
00358 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW
00359 { return l4_utcb_tcr_u(l4_utcb()); }
00360
00361 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW
00362 { return l4_utcb_exc_u(l4_utcb()); }
00363
00364 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW
00365 { l4_utcb_inherit_fpu_u(l4_utcb(), switch_on); }
00366
00367 L4_INLINE
00368 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t val, int pos,
00369 l4_utcb_t *utcb) L4_NOTHROW
00370 {
00371 union T
00372 {
00373 l4_kernel_clock_t t;
00374 l4_umword_t m[sizeof(l4_kernel_clock_t)/sizeof(
00375 l4_umword_t)];
00376 };
00376 l4_timeout_s to;
00377 to.t = 0x8000 | pos;
00378 ((union T*)(l4_utcb_br_u(utcb)->br + pos))->t = val;
00379 return to;
00380 }
00381
00382 L4_INLINE
00383 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t val, int pos)
00384 L4_NOTHROW
00385 { return l4_timeout_abs_u(val, pos, l4_utcb()); }
00386
00386 L4_INLINE unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW
00387 { return idx / (sizeof(l4_uint64_t) / sizeof(l4_umword_t)); }
00388
00389 __END_DECLS
00390

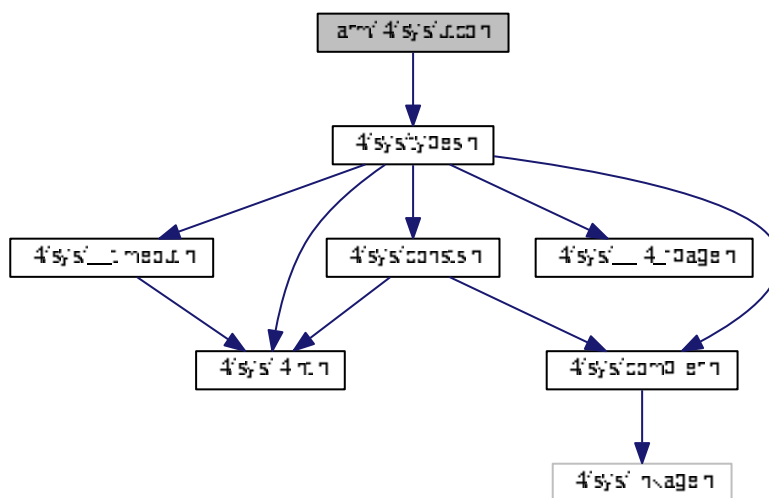
```

```
00391 #endif /* !_L4_SYS_UTCB_H */
```

## 15.371 arm/l4/sys/utcb.h File Reference

UTCB definitions for ARM.

```
#include <l4/sys/types.h>
Include dependency graph for utcb.h:
```



### Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Enumerations

- enum [L4\\_utcb\\_consts\\_arm](#)  
*UTCB constants for ARM.*

## Functions

- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW`  
Access function to get the program counter of the exception state.
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`  
Set the program counter register in the exception state.
- `l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW`  
Get the value out of an exception UTCB that describes the type of exception.
- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW`  
Check whether an exception IPC is a page fault.
- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW`  
Function to get the L4 style page fault address out of an exception.
- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW`  
Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.

### 15.371.1 Detailed Description

UTCB definitions for ARM.

Definition in file `utcb.h`.

## 15.372 utcb.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00025 #define __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00026
00027 #include <l4/sys/types.h>
00028
00038 typedef struct l4_exc_regs_t
00039 {
00040 l4_umword_t pfa;
00041 l4_umword_t err;
00043 l4_umword_t r[13];
00044 l4_umword_t sp;
00045 l4_umword_t ulr;
00046 l4_umword_t _dummy1;
00047 l4_umword_t pc;
00048 l4_umword_t cpsr;
00049 l4_umword_t tpidruro;
00050 } l4_exc_regs_t;
00051
00057 enum L4_utcb_consts_arm
00058 {
00059 L4_UTCB_EXCEPTION_REGS_SIZE = sizeof(
00060 l4_exc_regs_t) / sizeof(l4_umword_t),
00060 L4_UTCB_GENERIC_DATA_SIZE = 63,
00061 L4_UTCB_GENERIC_BUFFERS_SIZE = 58,
00062

```

```

00063 L4_UTCB_MSG_REGS_OFFSET = 0,
00064 L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(
00065 l4_umword_t),
00066 L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(
00067 l4_umword_t),
00068 L4_UTCB_INHERIT_FPU = 1UL << 24,
00069 L4_UTCB_OFFSET = 512,
00070 };
00071
00072 #include_next <l4/sys/utcb.h>
00073
00074 /*
00075 * =====
00076 * Implementations.
00077 */
00078
00079 #ifdef __GNUC__
00080 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00081 {
00082 register l4_utcb_t *utcb __asm__ ("r0");
00083 __asm__ ("mov lr, pc\n"
00084 "mov pc, #0xffffffff00\n"
00085 : "=r"(utcb) : : "lr");
00086 return utcb;
00087 }
00088 #endif
00089
00090 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00091 L4_NOTHROW
00092 {
00093 return u->pc;
00094 }
00095
00096 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00097 l4_addr_t pc) L4_NOTHROW
00098 {
00099 u->pc = pc;
00100 }
00101
00102 L4_INLINE l4_umword_t l4_utcb_exc_typeval(
00103 l4_exc_regs_t const *u) L4_NOTHROW
00104 {
00105 return u->err >> 26;
00106 }
00107
00108 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u)
00109 L4_NOTHROW
00110 {
00111 return ((u->err >> 26) & 0x30) == 0x20;
00112 }
00113
00114 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u)
00115 L4_NOTHROW
00116 {
00117 return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00118 }
00119
00120 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00121 l4_exc_regs_t const *u) L4_NOTHROW
00122 {
00123 return l4_utcb_exc_typeval(u) == 0x3e;
00124 }
00125
00126 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */

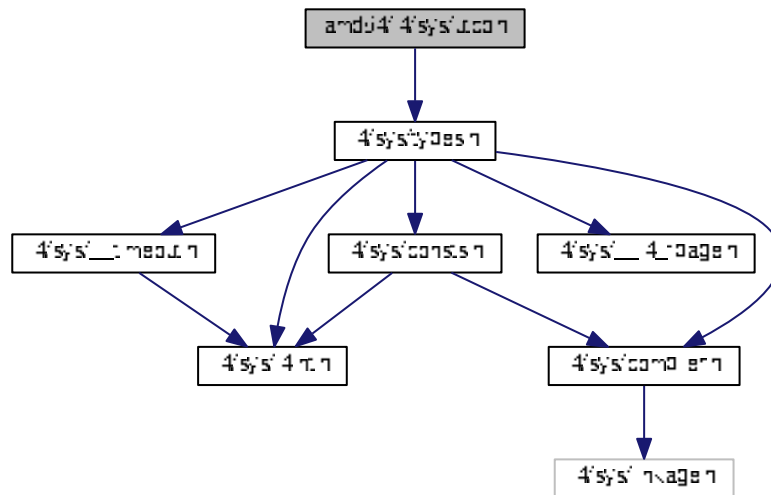
```

## 15.373 amd64/l4/sys/utcb.h File Reference

UTCB definitions for amd64.



```
#include <l4/sys/types.h>
Include dependency graph for utcb.h:
```



## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_amd64](#)  
*UTCB constants for AMD64.*

## Functions

- [l4\\_umword\\_t l4\\_utcb\\_exc\\_pc](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Access function to get the program counter of the exception state.*
- void [l4\\_utcb\\_exc\\_pc\\_set](#) ([l4\\_exc\\_regs\\_t](#) \*u, [l4\\_addr\\_t](#) pc) [L4\\_NOTHROW](#)  
*Set the program counter register in the exception state.*
- [l4\\_umword\\_t l4\\_utcb\\_exc\\_typeval](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Get the value out of an exception UTCB that describes the type of exception.*
- int [l4\\_utcb\\_exc\\_is\\_pf](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Check whether an exception IPC is a page fault.*
- [l4\\_addr\\_t l4\\_utcb\\_exc\\_pfa](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Function to get the L4 style page fault address out of an exception.*
- int [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).*

## 15.373.1 Detailed Description

UTCB definitions for amd64.

Definition in file [utcb.h](#).

## 15.374 utcb.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 /*****
00025 * #ifndef __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00026 * #define __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00027 *
00028 * #include <l4/sys/types.h>
00029 *
00039 * enum L4_utcb_consts_amd64
00040 * {
00041 * L4_UTCB_EXCEPTION_REGS_SIZE = 26,
00042 * L4_UTCB_GENERIC_DATA_SIZE = 63,
00043 * L4_UTCB_GENERIC_BUFFERS_SIZE = 58,
00044 *
00045 * L4_UTCB_MSG_REGS_OFFSET = 0,
00046 * L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(
00047 * l4_umword_t),
00048 * L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(
00049 * l4_umword_t),
00050 *
00051 * L4_UTCB_INHERIT_FPU = 1UL << 24,
00052 * L4_UTCB_OFFSET = 1024,
00053 * };
00054 *
00055 * typedef struct l4_exc_regs_t
00056 * {
00057 * l4_umword_t r15;
00058 * l4_umword_t r14;
00059 * l4_umword_t r13;
00060 * l4_umword_t r12;
00061 * l4_umword_t r11;
00062 * l4_umword_t r10;
00063 * l4_umword_t r9;
00064 * l4_umword_t r8;
00065 * l4_umword_t rdi;
00066 * l4_umword_t rsi;
00067 * l4_umword_t rbp;
00068 * l4_umword_t pfa;
00069 * l4_umword_t rbx;
00070 * l4_umword_t rdx;
00071 * l4_umword_t rcx;
00072 * l4_umword_t rax;
00073 * l4_umword_t trapno;
00074 * l4_umword_t err;
00075 * l4_umword_t ip;
00076 * l4_umword_t dummy1;
00077 * l4_umword_t flags;
00078 * l4_umword_t sp;
00079 * l4_umword_t ss;
00080 * l4_umword_t fs_base;
00081 * l4_umword_t gs_base;
00082 * l4_uint16_t ds, es, fs, gs;
00083 * } l4_exc_regs_t;
00084 */
00085

```

```

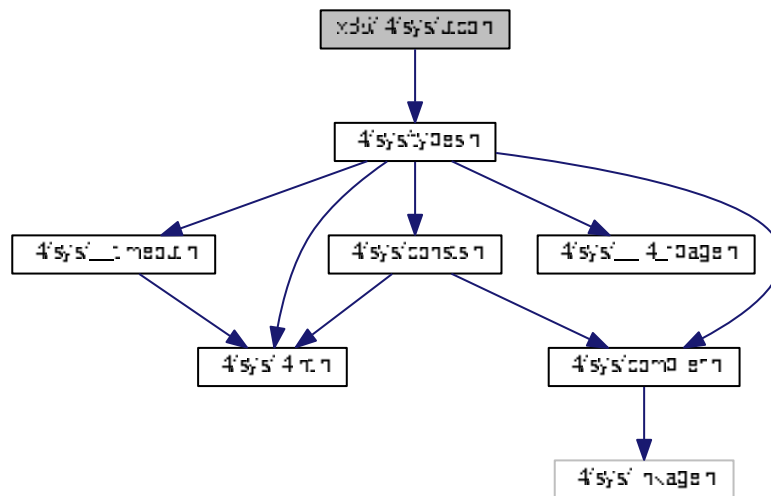
00088
00089 #include_next <l4/sys/utcb.h>
00090
00091 /*
00092 * =====
00093 * Implementations.
00094 */
00095
00096 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00097 {
00098 l4_utcb_t *res;
00099 __asm__ ("mov %%gs:0, %0 \n" : "=r"(res));
00100 return res;
00101 }
00102
00103 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00104 L4_NOTHROW
00105 {
00106 return u->ip;
00107 }
00108 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00109 l4_addr_t pc) L4_NOTHROW
00110 {
00111 u->ip = pc;
00112 }
00113 L4_INLINE l4_umword_t l4_utcb_exc_typeval(
00114 l4_exc_regs_t const *u) L4_NOTHROW
00115 {
00116 return u->trapno;
00117 }
00118 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u)
00119 L4_NOTHROW
00120 {
00121 return u->trapno == 14;
00122 }
00123 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u)
00124 L4_NOTHROW
00125 {
00126 return (u->pfa & ~7UL) | (u->err & 2);
00127 }
00128 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00129 l4_exc_regs_t const *u) L4_NOTHROW
00130 {
00131 return l4_utcb_exc_typeval(u) == 0xff;
00132 }
00133 #endif /* ! __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__ */

```

## 15.375 x86/l4/sys/utcb.h File Reference

UTCB definitions for X86.

```
#include <l4/sys/types.h>
Include dependency graph for utcb.h:
```



## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_x86](#) {  
[L4\\_UTCB\\_EXCEPTION\\_REGS\\_SIZE](#) = 19, [L4\\_UTCB\\_GENERIC\\_DATA\\_SIZE](#) = 63, [L4\\_UTCB\\_GENERIC\\_BUFFERS\\_SIZE](#) = 58, [L4\\_UTCB\\_MSG\\_REGS\\_OFFSET](#) = 0,  
[L4\\_UTCB\\_BUF\\_REGS\\_OFFSET](#) = 64 \* sizeof([l4\\_umword\\_t](#)), [L4\\_UTCB\\_THREAD\\_REGS\\_OFFSET](#) = 123 \* sizeof([l4\\_umword\\_t](#)), [L4\\_UTCB\\_INHERIT\\_FPU](#) = 1UL << 24, [L4\\_UTCB\\_OFFSET](#) = 512 }  
*UTCB constants for x86.*

## Functions

- [l4\\_umword\\_t](#) [l4\\_utcb\\_exc\\_pc](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
*Access function to get the program counter of the exception state.*
- void [l4\\_utcb\\_exc\\_pc\\_set](#) ([l4\\_exc\\_regs\\_t](#) \*u, [l4\\_addr\\_t](#) pc) [L4\\_NOTHROW](#)  
*Set the program counter register in the exception state.*
- [l4\\_umword\\_t](#) [l4\\_utcb\\_exc\\_typeval](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)

Get the value out of an exception UTCB that describes the type of exception.

- int [l4\\_utcb\\_exc\\_is\\_pf](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)

Check whether an exception IPC is a page fault.

- [l4\\_addr\\_t](#) [l4\\_utcb\\_exc\\_pfa](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)

Function to get the [L4](#) style page fault address out of an exception.

- int [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)

Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).

### 15.375.1 Detailed Description

UTCB definitions for X86.

Definition in file [utcb.h](#).

## 15.376 utcb.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 /*****
00026 #ifndef __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00027 #define __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00028
00029 #include <l4/sys/types.h>
00030
00041 enum L4_utcb_consts_x86
00042 {
00044 L4_UTCB_EXCEPTION_REGS_SIZE = 19,
00045
00047 L4_UTCB_GENERIC_DATA_SIZE = 63,
00048
00050 L4_UTCB_GENERIC_BUFFERS_SIZE = 58,
00051
00053 L4_UTCB_MSG_REGS_OFFSET = 0,
00054
00056 L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(
00057 l4_umword_t),
00059 L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(
00060 l4_umword_t),
00062 L4_UTCB_INHERIT_FPU = 1UL << 24,
00063
00065 L4_UTCB_OFFSET = 512,
00066 };
00067
00072 typedef struct l4_exc_regs_t
00073 {
00074 l4_umword_t es;
00075 l4_umword_t ds;
00076 l4_umword_t gs;
00077 l4_umword_t fs;
00079 l4_umword_t edi;
00080 l4_umword_t esi;
00081 l4_umword_t ebp;

```

```

00082 l4_umword_t pfa;
00083 l4_umword_t ebx;
00084 l4_umword_t edx;
00085 l4_umword_t ecx;
00086 l4_umword_t eax;
00088 l4_umword_t trapno;
00089 l4_umword_t err;
00091 l4_umword_t ip;
00092 l4_umword_t dummy1;
00093 l4_umword_t flags;
00094 l4_umword_t sp;
00095 l4_umword_t ss;
00096 } l4_exc_regs_t;
00097
00098 #include_next <l4/sys/utcb.h>
00099
00100 /*
00101 * =====
00102 * Implementations.
00103 */
00104
00105 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00106 {
00107 l4_utcb_t *utcb;
00108 __asm__ ("mov %%fs:0, %0" : "=r" (utcb));
00109 return utcb;
00110 }
00111
00112 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00113 L4_NOTHROW
00114 {
00115 return u->ip;
00116 }
00117 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00118 l4_addr_t pc) L4_NOTHROW
00119 {
00120 u->ip = pc;
00121 }
00122 L4_INLINE void l4_utcb_exc_sp_set(l4_exc_regs_t *u, l4_addr_t
00123 sp) L4_NOTHROW
00124 {
00125 u->sp = sp;
00126 }
00127 L4_INLINE l4_umword_t l4_utcb_exc_typeval(
00128 l4_exc_regs_t const *u) L4_NOTHROW
00129 {
00130 return u->trapno;
00131 }
00132 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u)
00133 L4_NOTHROW
00134 {
00135 return u->trapno == 14;
00136 }
00137 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u)
00138 L4_NOTHROW
00139 {
00140 return (u->pfa & ~7UL) | (u->err & 2);
00141 }
00142 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00143 l4_exc_regs_t const *u) L4_NOTHROW
00144 {
00145 return l4_utcb_exc_typeval(u) == 0xff;
00146 }
00147 #endif /* ! __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__ */

```

## 15.377 l4/sys/vcon File Reference

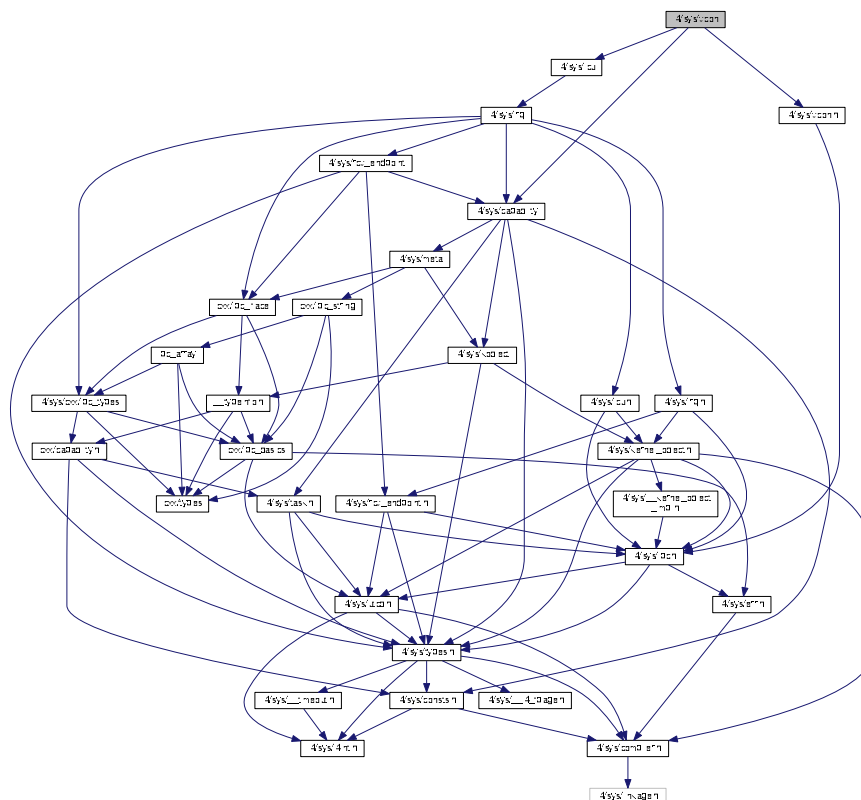
Virtual console interface.

```

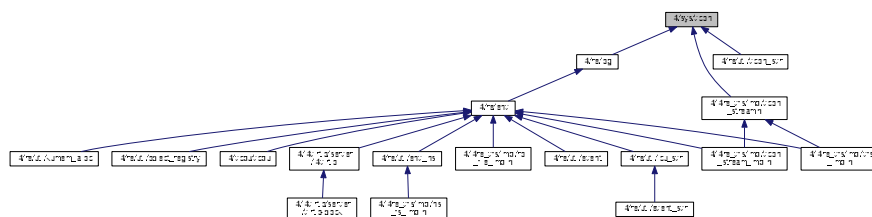
#include <l4/sys/icu>
#include <l4/sys/vcon.h>

```

```
#include <l4/sys/capability>
Include dependency graph for vcon:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Vcon`  
*C++ L4 Vcon interface.*

## Namespaces

- L4  
*L4 low-level kernel interface.*

### 15.377.1 Detailed Description

Virtual console interface.

Definition in file [vcon](#).

## 15.378 vcon

```

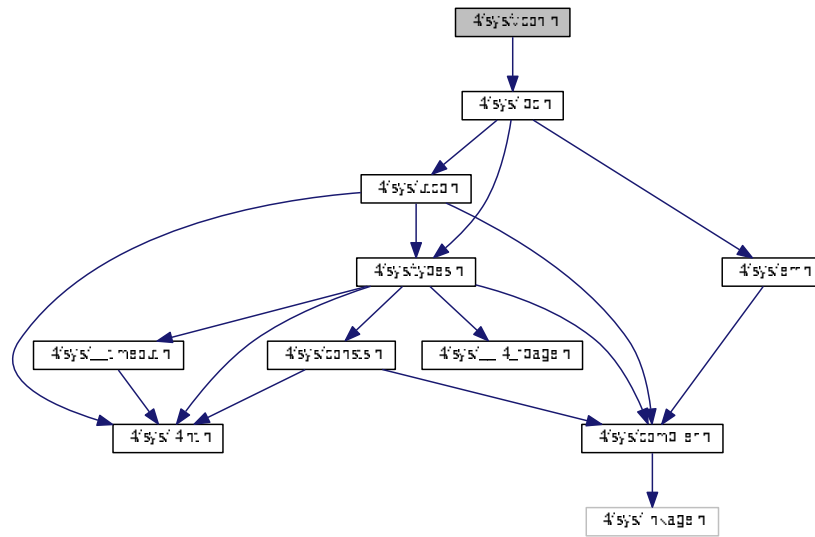
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006 * economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 #include <l4/sys/icu>
00024 #include <l4/sys/vcon.h>
00025 #include <l4/sys/capability>
00026
00027 namespace L4 {
00028
00029 class Vcon :
00030 public Kobject_t<Vcon, Icu, L4_PROTO_LOG>
00031 {
00032 public:
00033 l4_msgtag_t
00034 send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
00035 { return l4_vcon_send_u(cap(), buf, size, utcb); }
00036
00037 long
00038 write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
00039 { return l4_vcon_write_u(cap(), buf, size, utcb); }
00040
00041 int
00042 read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
00043 { return l4_vcon_read_u(cap(), buf, size, utcb); }
00044
00045 int
00046 read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb =
00047 l4_utcb()) const throw()
00048 { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }
00049
00050 l4_msgtag_t
00051 set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb =
00052 l4_utcb()) const throw()
00053 { return l4_vcon_set_attr_u(cap(), attr, utcb); }
00054
00055 l4_msgtag_t
00056 get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb =
00057 l4_utcb()) const throw()
00058 { return l4_vcon_get_attr_u(cap(), attr, utcb); }
00059
00060 typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
00061 };
00062
00063 }
```

## 15.379 l4/sys/vcon.h File Reference

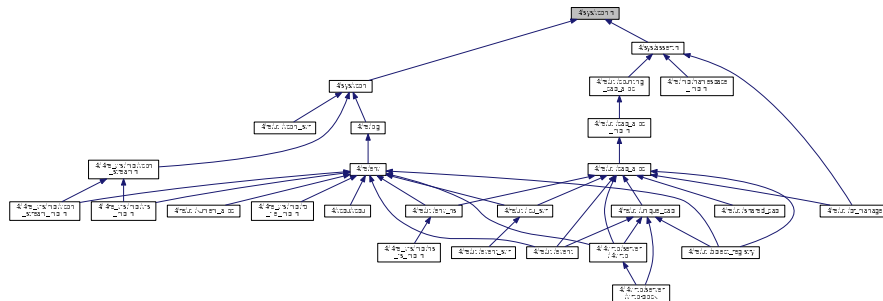
Virtual console interface.



```
#include <linux/sys/ipc.h>
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4_vcon_attr_t`  
*Vcon attribute structure.*

## Typedefs

- typedef struct l4\_vcon\_attr\_t l4\_vcon\_attr\_t  
*Vcon attribute structure.*

## Enumerations

- enum [L4\\_vcon\\_size\\_consts](#) { [L4\\_VCON\\_WRITE\\_SIZE](#) = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 2) \* sizeof(l4\_umword\_t), [L4\\_VCON\\_READ\\_SIZE](#) = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 1) \* sizeof(l4\_umword\_t) }  
Size constants.
- enum [L4\\_vcon\\_read\\_flags](#) { [L4\\_VCON\\_READ\\_SIZE\\_MASK](#) = 0x3ffffff, [L4\\_VCON\\_READ\\_STAT\\_BREAK](#) = 1 << 30, [L4\\_VCON\\_READ\\_STAT\\_DONE](#) = 1 << 31 }  
Vcon read flags.
- enum [L4\\_vcon\\_i\\_flags](#) { [L4\\_VCON\\_INLCR](#) = 000100, [L4\\_VCON\\_IGNCR](#) = 000200, [L4\\_VCON\\_ICRNL](#) = 000400 }  
Input flags.
- enum [L4\\_vcon\\_o\\_flags](#) { [L4\\_VCON\\_ONLCR](#) = 000004, [L4\\_VCON\\_OCRNL](#) = 000010, [L4\\_VCON\\_ONLRET](#) = 000040 }  
Output flags.
- enum [L4\\_vcon\\_l\\_flags](#) { [L4\\_VCON\\_ICANON](#) = 000002, [L4\\_VCON\\_ECHO](#) = 000010 }  
Local flags.
- enum [L4\\_vcon\\_ops](#) { [L4\\_VCON\\_WRITE\\_OP](#) = 0UL, [L4\\_VCON\\_READ\\_OP](#) = 1UL, [L4\\_VCON\\_SET\\_ATTR\\_OP](#) = 2UL, [L4\\_VCON\\_GET\\_ATTR\\_OP](#) = 3UL }  
Operations on vcon objects.

## Functions

- [l4\\_msgtag\\_t l4\\_vcon\\_send](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size) [L4\\_NOTHROW](#)  
Send data to virtual console.
- [l4\\_msgtag\\_t l4\\_vcon\\_send\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
Send data to *this* virtual console.
- long [l4\\_vcon\\_write](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size) [L4\\_NOTHROW](#)  
Write data to virtual console.
- long [l4\\_vcon\\_write\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
Write data to *this* virtual console.
- int [l4\\_vcon\\_read](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size) [L4\\_NOTHROW](#)  
Read data from virtual console.
- int [l4\\_vcon\\_read\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
Read data from *this* virtual console.
- int [l4\\_vcon\\_read\\_with\\_flags](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size) [L4\\_NOTHROW](#)  
Read data from virtual console, extended version including flags.
- [l4\\_msgtag\\_t l4\\_vcon\\_set\\_attr](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) const \*attr) [L4\\_NOTHROW](#)  
Set attributes of a Vcon.
- [l4\\_msgtag\\_t l4\\_vcon\\_set\\_attr\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) const \*attr, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
Set the attributes of *this* virtual console.
- [l4\\_msgtag\\_t l4\\_vcon\\_get\\_attr](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) \*attr) [L4\\_NOTHROW](#)  
Get attributes of a Vcon.
- [l4\\_msgtag\\_t l4\\_vcon\\_get\\_attr\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) \*attr, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
Get attributes of *this* virtual console.

### 15.379.1 Detailed Description

Virtual console interface.

Definition in file [vcon.h](#).

## 15.379.2 Enumeration Type Documentation

### 15.379.2.1 L4\_vcon\_read\_flags

enum [L4\\_vcon\\_read\\_flags](#)

Vcon read flags.

#### Enumerator

|                                         |                       |
|-----------------------------------------|-----------------------|
| <a href="#">L4_VCON_READ_SIZE_MASK</a>  | Size mask.            |
| <a href="#">L4_VCON_READ_STAT_BREAK</a> | Break condition flag. |
| <a href="#">L4_VCON_READ_STAT_DONE</a>  | Done condition flag.  |

Definition at line [167](#) of file [vcon.h](#).

## 15.380 vcon.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/ipc.h>
00027
00056 L4_INLINE l4_msgtag_t
00057 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size)
00058 L4_NOTHROW;
00059
00065 L4_INLINE l4_msgtag_t
00066 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
00067 l4_utcb_t *utcb) L4_NOTHROW;
00068
00079 L4_INLINE long
00080 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size)
00081 L4_NOTHROW;
00082
00088 L4_INLINE long
00089 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
00090 l4_utcb_t *utcb) L4_NOTHROW;
00091
00095 enum L4_vcon_size_consts
00096 {
00098 L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(
00099 l4_umword_t),
00100 L4_VCON_READ_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(
00101 l4_umword_t),
00102 };

```

```

00102
00118 L4_INLINE int
00119 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size)
 L4_NOTHROW;
00120
00127 L4_INLINE int
00128 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size,
 l4_utcb_t *utcb) L4_NOTHROW;
00129
00154 L4_INLINE int
00155 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size)
 L4_NOTHROW;
00156
00160 L4_INLINE int
00161 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
 l4_utcb_t *utcb) L4_NOTHROW;
00162
00163
00167 enum L4_vcon_read_flags
00168 {
00169 L4_VCON_READ_SIZE_MASK = 0x3fffffff,
00170 L4_VCON_READ_STAT_BREAK = 1 << 30,
00171 L4_VCON_READ_STAT_DONE = 1 << 31,
00172 };
00173
00178 typedef struct l4_vcon_attr_t
00179 {
00180 l4_umword_t i_flags;
00181 l4_umword_t o_flags;
00182 l4_umword_t l_flags;
00183 } l4_vcon_attr_t;
00184
00189 enum L4_vcon_i_flags
00190 {
00191 L4_VCON_INLCR = 000100,
00192 L4_VCON_IGNCR = 000200,
00193 L4_VCON_ICRNL = 000400,
00194 };
00195
00200 enum L4_vcon_o_flags
00201 {
00202 L4_VCON_ONLCR = 000004,
00203 L4_VCON_OCRNL = 000010,
00204 L4_VCON_ONLRET = 000040,
00205 };
00206
00211 enum L4_vcon_l_flags
00212 {
00213 L4_VCON_ICANON = 000002,
00214 L4_VCON_ECHO = 000010,
00215 };
00216
00225 L4_INLINE l4_msgtag_t
00226 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr)
 L4_NOTHROW;
00227
00234 L4_INLINE l4_msgtag_t
00235 l4_vcon_set_attr_u(l4_cap_idx_t vcon,
 l4_vcon_attr_t const *attr,
00236 l4_utcb_t *utcb) L4_NOTHROW;
00237
00246 L4_INLINE l4_msgtag_t
00247 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr)
 L4_NOTHROW;
00248
00255 L4_INLINE l4_msgtag_t
00256 l4_vcon_get_attr_u(l4_cap_idx_t vcon,
 l4_vcon_attr_t *attr,
00257 l4_utcb_t *utcb) L4_NOTHROW;
00258
00259
00264 enum L4_vcon_ops
00265 {
00266 L4_VCON_WRITE_OP = 0UL,
00267 L4_VCON_READ_OP = 1UL,
00268 L4_VCON_SET_ATTR_OP = 2UL,
00269 L4_VCON_GET_ATTR_OP = 3UL,
00270 };
00271
00272 /***** Implementations *****/
00273
00274 L4_INLINE l4_msgtag_t
00275 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
 l4_utcb_t *utcb) L4_NOTHROW
00276 {
00277 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00278 mr->mr[0] = L4_VCON_WRITE_OP;
00279 mr->mr[1] = size;

```

```

00280 __builtin_memcpy(&mr->mr[2], buf, size);
00281 return l4_ipc_send(vcon, utcb,
00282 l4_msgtag(L4_PROTO_LOG,
00283 2 + (size + sizeof(l4_umword_t) - 1) / sizeof(
00284 l4_umword_t),
00285 0, L4_MSGTAG_SCHEDULE),
00286);
00287
00288 L4_INLINE l4_msgtag_t
00289 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00290 {
00291 return l4_vcon_send_u(vcon, buf, size, l4_utcb());
00292 }
00293
00294 L4_INLINE long
00295 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
00296 l4_utcb_t *utcb) L4_NOTHROW
00297 {
00298 l4_msgtag_t t;
00299 if (size > L4_VCON_WRITE_SIZE)
00300 size = L4_VCON_WRITE_SIZE;
00301 t = l4_vcon_send_u(vcon, buf, size, utcb);
00302 if (l4_msgtag_has_error(t))
00303 return l4_error(t);
00304 return (long) size;
00305 }
00306
00307 L4_INLINE long
00308 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00309 {
00310 return l4_vcon_write_u(vcon, buf, size, l4_utcb());
00311 }
00312
00313 L4_INLINE int
00314 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00315 l4_utcb_t *utcb) L4_NOTHROW
00316 {
00317 int ret;
00318 unsigned r;
00319 l4_msg_regs_t *mr;
00320 mr = l4_utcb_mr_u(utcb);
00321 mr->mr[0] = (size << 16) | L4_VCON_READ_OP;
00322 ret = l4_error_u(l4_ipc_call(vcon, utcb,
00323 l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00324 L4_IPC_NEVER),
00325 utcb);
00326 if (ret < 0)
00327 return ret;
00328 r = mr->mr[0] & L4_VCON_READ_SIZE_MASK;
00329 if (!(mr->mr[0] & L4_VCON_READ_STAT_DONE)) // !eof
00330 ret = size + 1;
00331 else if (r < size)
00332 ret = r;
00333 else
00334 ret = size;
00335 if (L4_LIKELY(buf != NULL))
00336 __builtin_memcpy(buf, &mr->mr[1], r < size ? r : size);
00337 return ret | (mr->mr[0] & ~(L4_VCON_READ_STAT_DONE |
00338 L4_VCON_READ_SIZE_MASK));
00339 }
00340
00341 L4_INLINE int
00342 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size)
00343 L4_NOTHROW
00344 {
00345 return l4_vcon_read_with_flags_u(vcon, buf, size, l4_utcb());
00346 }
00347
00348 L4_INLINE int
00349 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00350 l4_utcb_t *utcb) L4_NOTHROW
00351 {
00352 int r = l4_vcon_read_with_flags_u(vcon, buf, size, utcb);
00353 if (r < 0)
00354 return r;
00355 return r & L4_VCON_READ_SIZE_MASK;
00356 }

```

```

00362 }
00363
00364 L4_INLINE int
00365 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00366 {
00367 return l4_vcon_read_u(vcon, buf, size, l4_utcb());
00368 }
00369
00370 L4_INLINE l4_msgtag_t
00371 l4_vcon_set_attr_u(l4_cap_idx_t vcon,
00372 l4_vcon_attr_t const *attr,
00373 l4_utcb_t *utcb) L4_NOTHROW
00374 {
00375 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00376 mr->mr[0] = L4_VCON_SET_ATTR_OP;
00377 __builtin_memcpy(&mr->mr[1], attr, sizeof(*attr));
00378
00379 return l4_ipc_call(vcon, utcb,
00380 l4_msgtag(L4_PROTO_LOG, 4, 0, 0),
00381 L4_IPC_NEVER);
00382 }
00383
00384 L4_INLINE l4_msgtag_t
00385 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr)
00386 L4_NOTHROW
00387 {
00388 return l4_vcon_set_attr_u(vcon, attr, l4_utcb());
00389 }
00390 L4_INLINE l4_msgtag_t
00391 l4_vcon_get_attr_u(l4_cap_idx_t vcon,
00392 l4_vcon_attr_t *attr,
00393 l4_utcb_t *utcb) L4_NOTHROW
00394 {
00395 l4_msgtag_t res;
00396 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00397 mr->mr[0] = L4_VCON_GET_ATTR_OP;
00398
00399 res = l4_ipc_call(vcon, utcb,
00400 l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00401 L4_IPC_NEVER);
00402 if (l4_error_u(res, utcb) >= 0)
00403 __builtin_memcpy(attr, &mr->mr[1], sizeof(*attr));
00404
00405 return res;
00406 }
00407
00408 L4_INLINE l4_msgtag_t
00409 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr)
00410 L4_NOTHROW
00411 {
00412 return l4_vcon_get_attr_u(vcon, attr, l4_utcb());
00413 }

```

## 15.381 l4/sys/vhw.h File Reference

Descriptors for virtual hardware (under UX).

```

#include <l4/sys/types.h>
#include <l4/sys/kip.h>

```



## 15.382 vhw.h

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 /*****
00026 #ifndef _L4_SYS_VHW_H
00027 #define _L4_SYS_VHW_H
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/kip.h>
00031
00044 enum l4_vhw_entry_type {
00045 L4_TYPE_VHW_NONE,
00046 L4_TYPE_VHW_FRAMEBUFFER,
00047 L4_TYPE_VHW_INPUT,
00048 L4_TYPE_VHW_NET,
00049 };
00050
00055 struct l4_vhw_entry {
00056 enum l4_vhw_entry_type type;
00057 l4_uint32_t provider_pid;
00059 l4_addr_t mem_start;
00060 l4_addr_t mem_size;
00062 l4_uint32_t irq_no;
00063 l4_uint32_t fd;
00064 };
00065
00070 struct l4_vhw_descriptor {
00071 l4_uint32_t magic;
00072 l4_uint8_t version;
00073 l4_uint8_t count;
00074 l4_uint8_t pad1;
00075 l4_uint8_t pad2;
00077 struct l4_vhw_entry descs[];
00078 };
00079
00080 enum {
00081 L4_VHW_MAGIC = 0x56687765,
00082 };
00083
00084 static inline struct l4_vhw_descriptor *
00085 l4_vhw_get(l4_kernel_info_t *kip) L4_NOTHROW
00086 {
00087 struct l4_vhw_descriptor *v
00088 = (struct l4_vhw_descriptor *)(((unsigned long)kip) + kip->vhw_offset);
00089
00090 if (v->magic == L4_VHW_MAGIC)
00091 return v;
00092
00093 return NULL;
00094 }
00095
00096 static inline struct l4_vhw_entry *
00097 l4_vhw_get_entry(struct l4_vhw_descriptor *v, int entry)
00098 L4_NOTHROW
00099 {
00100 return v->descs + entry;
00101 }
00102
00103 static inline struct l4_vhw_entry *
00104 l4_vhw_get_entry_type(struct l4_vhw_descriptor *v, enum
00105 l4_vhw_entry_type t) L4_NOTHROW
00106 {
00107 int i;
00108 struct l4_vhw_entry *e = v->descs;
00109
00110 for (i = 0; i < v->count; i++, e++)
00111 if (e->type == t)
00112 return e;

```



```

00111
00112 return NULL;
00113 }
00114
00115 #endif /* ! _L4_SYS_VHW_H */

```

## 15.383 l4/sys/vm File Reference

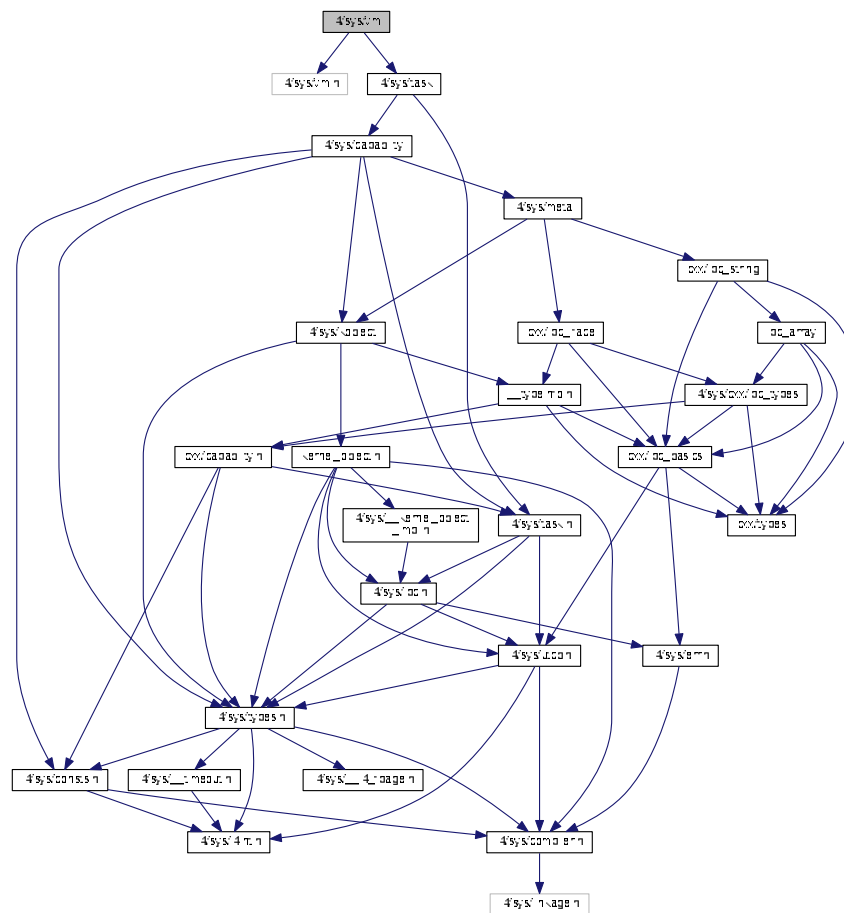
Virtualization interface.

```

#include <l4/sys/vm.h>
#include <l4/sys/task>

```

Include dependency graph for vm:



### Data Structures

- class [L4::Vm](#)  
*Virtual machine.*

### Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 15.383.1 Detailed Description

Virtualization interface.

Definition in file [vm](#).

## 15.384 vm

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/vm.h>
00028 #include <l4/sys/task>
00029
00030 namespace L4 {
00031
00040 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00041 {
00042 protected:
00043 Vm();
00044
00045 private:
00046 Vm(Vm const &);
00047 void operator = (Vm const &);
00048 };
00049
00050 };

```

## 15.385 l4/util/alloc.h File Reference

Allocator using a bit-array.

```

#include <l4/sys/l4int.h>
#include <l4/util/bitops.h>
#include <l4/sys/compiler.h>

```



```

00023
00024 typedef struct {
00025 int base, count, next_elem;
00026 l4_umword_t *bits;
00027 } l4util_alloc_t;
00028
00029 #define L4UTIL_ALLOC_BITS_SIZE (8 * sizeof(l4_umword_t))
00030
00031 L4_CV l4util_alloc_t *l4util_alloc_init(int count, int base);
00032 L4_CV int l4util_alloc_avail(l4util_alloc_t *alloc, int elem);
00033 L4_CV int l4util_alloc_occupy(l4util_alloc_t *alloc, int elem);
00034 L4_CV int l4util_alloc_alloc(l4util_alloc_t *alloc);
00035 L4_CV int l4util_alloc_free(l4util_alloc_t *alloc, int elem);
00036
00037 EXTERN_C_END
00038 #endif

```

## 15.387 l4/cxx/alloc.h File Reference

Alloc list.

### Data Structures

- class [L4::Alloc\\_list](#)  
*A simple list-based allocator.*

### Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

## 15.387.1 Detailed Description

Alloc list.

Definition in file [alloc.h](#).

## 15.388 alloc.h

```

00001
00005 /*
00006 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024

```

```

00025 namespace L4 {
00026
00031 class Alloc_list
00032 {
00033 public:
00034 Alloc_list() : _free(0) {}
00035 Alloc_list(void *blk, unsigned long size) : _free(0)
00036 { free(blk, size); }
00037
00038 void free(void *blk, unsigned long size);
00039 void *alloc(unsigned long size);
00040
00041 private:
00042 struct Elem
00043 {
00044 Elem *next;
00045 unsigned long size;
00046 };
00047
00048 Elem *_free;
00049 };
00050 }

```

## 15.389 l4/util/assert.h File Reference

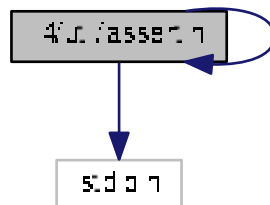
Some useful assert-style macros.

```

#include <stdio.h>
#include <assert.h>

```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



### 15.389.1 Detailed Description

Some useful assert-style macros.

#### Date

09/2009

#### Author

Bjoern Doebel [doebel@tudos.org](mailto:doebel@tudos.org)

Definition in file [assert.h](#).

## 15.390 assert.h

```

00001 /*****
00009 */
00010 * (c) 2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #pragma once
00019
00020 #ifndef NDEBUG
00021
00022 #define DO_NOTHING do {} while (0)
00023 #define ASSERT_VALID(c) DO_NOTHING
00024 #define ASSERT_EQUAL(a,b) DO_NOTHING
00025 #define ASSERT_NOT_EQUAL(a,b) DO_NOTHING
00026 #define ASSERT_LOWER_EQ(a,b) DO_NOTHING
00027 #define ASSERT_GREATER_EQ(a,b) DO_NOTHING
00028 #define ASSERT_BETWEEN(a,b,c) DO_NOTHING
00029 #define ASSERT_IPC_OK(i) DO_NOTHING
00030 #define ASSERT_OK(e) do { (void)e; } while (0)
00031 #define ASSERT_NOT_NULL(p) DO_NOTHING
00032 #ifndef assert
00033 #define assert(cond) DO_NOTHING
00034 #endif
00035
00036 #else // NDEBUG
00037
00038 #ifndef ASSERT_PRINTF
00039 #include <stdio.h>
00040 #define ASSERT_PRINTF printf
00041 #endif
00042 #ifndef ASSERT_ASSERT
00043 #include <assert.h>
00044 #define ASSERT_ASSERT(x) assert(x)
00045 #endif
00046
00047 #define ASSERT_VALID(cap) \
00048 do { \
00049 typeof(cap) _cap = cap; \
00050 if (!l4_is_invalid_cap(_cap)) { \
00051 ASSERT_PRINTF("%s: Cap invalid.\n", __func__); \
00052 ASSERT_ASSERT(!l4_is_invalid_cap(_cap)); \
00053 } \
00054 } while (0)
00055
00056
00057 #define ASSERT_EQUAL(a, b) \
00058 do { \
00059 typeof(a) _a = a; \
00060 typeof(b) _b = b; \
00061 if (_a != _b) { \
00062 ASSERT_PRINTF("%s:\n", __func__); \
00063 ASSERT_PRINTF(" "#a" (%lx) != "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00064 ASSERT_ASSERT(_a == _b); \

```

```

00065 } \
00066 } while (0)
00067
00068
00069 #define ASSERT_NOT_EQUAL(a, b) \
00070 do { \
00071 typeof(a) _a = a; \
00072 typeof(b) _b = b; \
00073 if (_a == _b) { \
00074 ASSERT_PRINTF("%s:\n", __func__); \
00075 ASSERT_PRINTF(" "#a" (%lx) == "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00076 ASSERT_ASSERT(_a != _b); \
00077 } \
00078 } while (0)
00079
00080
00081 #define ASSERT_LOWER_EQ(val, max) \
00082 do { \
00083 typeof(val) _val = val; \
00084 typeof(max) _max = max; \
00085 if (_val > _max) { \
00086 ASSERT_PRINTF("%s:\n", __func__); \
00087 ASSERT_PRINTF(" "#val" (%lx) > "#max" (%lx)\n", (unsigned long)_val, (unsigned long)_max); \
00088 ASSERT_ASSERT(_val <= _max); \
00089 } \
00090 } while (0)
00091
00092
00093 #define ASSERT_GREATER_EQ(val, min) \
00094 do { \
00095 typeof(val) _val = val; \
00096 typeof(min) _min = min; \
00097 if (_val < _min) { \
00098 ASSERT_PRINTF("%s:\n", __func__); \
00099 ASSERT_PRINTF(" "#val" (%lx) < "#min" (%lx)\n", (unsigned long)_val, (unsigned long)_min); \
00100 ASSERT_ASSERT(_val >= _min); \
00101 } \
00102 } while (0)
00103
00104
00105 #define ASSERT_BETWEEN(val, min, max) \
00106 ASSERT_LOWER_EQ((val), (max)); \
00107 ASSERT_GREATER_EQ((val), (min));
00108
00109
00110 #define ASSERT_IPC_OK(msgtag) \
00111 do { \
00112 int _r = l4_ipc_error(msgtag, l4_utcb()); \
00113 if (_r) { \
00114 ASSERT_PRINTF("%s: IPC Error: %lx\n", __func__, _r); \
00115 ASSERT_ASSERT(_r == 0); \
00116 } \
00117 } while (0)
00118
00119 #define ASSERT_OK(val) ASSERT_EQUAL((val), 0)
00120 #define ASSERT_NOT_NULL(ptr) ASSERT_NOT_EQUAL((ptr), (void *)0)
00121
00122 #endif // NDEBUG

```

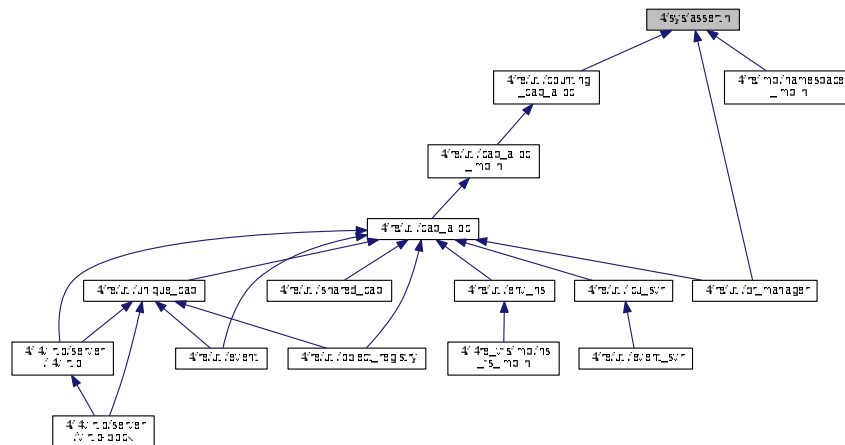
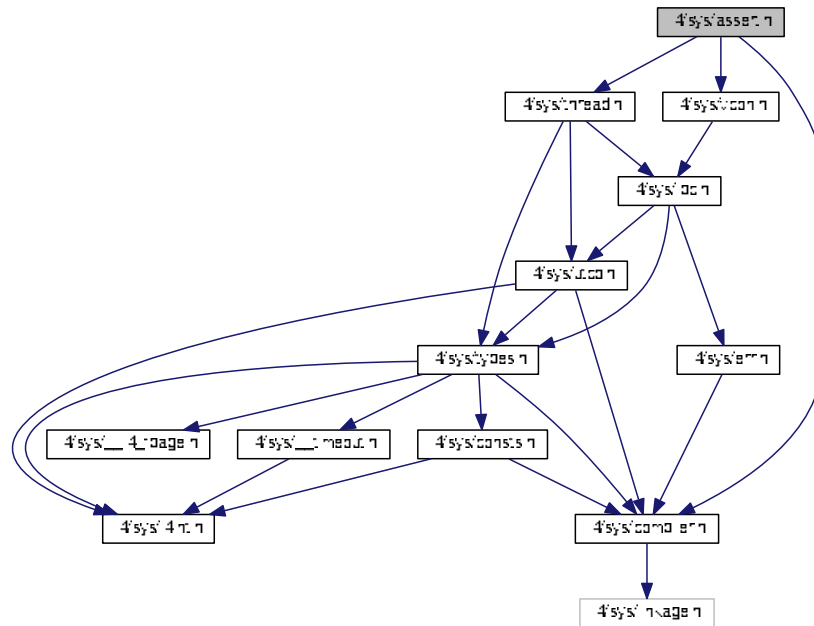
## 15.391 l4/sys/assert.h File Reference

Low-level assert implementation.

```

#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
#include <l4/sys/vcon.h>

```



- `#define l4_assert(expr)`  
*Low-level assert.*

### Low-level assert implementation.

### Low-level assert implementation.

Generated for L4Re by Doxygen



## 15.391.2 Macro Definition Documentation

### 15.391.2.1 l4\_assert

```
#define l4_assert(
 expr)
```

#### Value:

```
l4_assert_fn(expr, __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
 L4_stringify(expr) "\" failed.\n")
```

Low-level assert.

#### Parameters

|             |                                              |
|-------------|----------------------------------------------|
| <i>expr</i> | Expression to be evaluate for the assertion. |
|-------------|----------------------------------------------|

This assertion is a low-level implementation that directly uses kernel primitives. Only use `l4_assert()` when the standard `assert()` functionality is not available.

Definition at line 43 of file `assert.h`.

Referenced by `L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free()`, `L4Re::Util::Counting_cap_alloc< C↵OUNTERTYPE >::release()`, and `L4Re::Util::Br_manager::set_rcv_cap_flags()`.

## 15.392 assert.h

```
00001
00005 /*
00006 * (c) 2015 Adam Lackorzynski <adam@l4re.org>
00007 *
00008 * This file is part of L4Re and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 #ifdef NDEBUG
00024
00025 #define l4_assert(x) do { } while (0)
00026 #define l4_check(x) do { (void)(x); } while (0)
00027
00028 #else
00029
00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/thread.h>
00032 #include <l4/sys/vcon.h>
00033
00043 #define l4_assert(expr) \
```

```

00044 l4_assert_fn(expr, __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
00045 L4_stringify(expr) "\" failed.\n")
00046
00047 #define l4_check(expr) l4_assert(expr)
00048
00052 L4_ALWAYS_INLINE
00053 void l4_assert_fn(bool expr, const char *text) L4_NOTHROW;
00054
00058 L4_INLINE L4_NORETURN
00059 void l4_assert_abort(const char *text) L4_NOTHROW;
00060
00061
00062 /* IMPLEMENTATION -----*/
00063
00064 L4_INLINE L4_NORETURN
00065 void l4_assert_abort(const char *text) L4_NOTHROW
00066 {
00067 l4_vcon_write(L4_BASE_LOG_CAP, text, __builtin_strlen(text));
00068 for (;;)
00069 l4_thread_ex_regs(L4_INVALID_CAP, ~0UL, ~0UL,
00070 L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
00071 }
00072
00073 L4_ALWAYS_INLINE
00074 void l4_assert_fn(bool expr, const char *text) L4_NOTHROW
00075 {
00076 if (L4_LIKELY(expr))
00077 return;
00078 l4_assert_abort(text);
00079 }
00080 }
00081
00082 #endif /* NDEBUG */

```

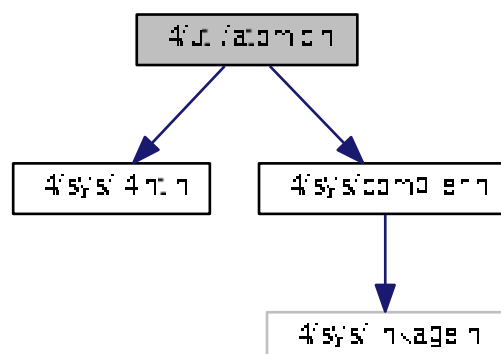
## 15.393 l4/util/atomic.h File Reference

atomic operations header and generic implementations

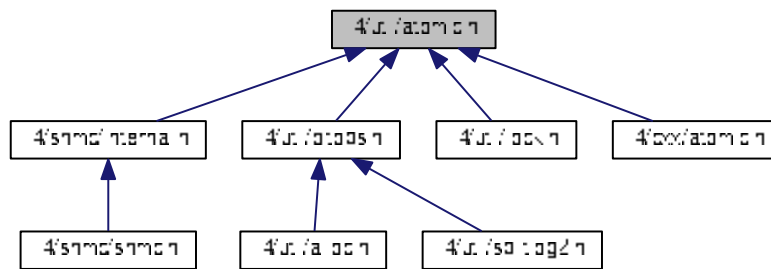
```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
Include dependency graph for atomic.h:

```



This graph shows which files directly or indirectly include this file:



## Functions

- `int l4util_cmpxchg64 (volatile l4_uint64_t *dest, l4_uint64_t cmp_val, l4_uint64_t new_val)`  
*Atomic compare and exchange (64 bit version)*
- `int l4util_cmpxchg32 (volatile l4_uint32_t *dest, l4_uint32_t cmp_val, l4_uint32_t new_val)`  
*Atomic compare and exchange (32 bit version)*
- `int l4util_cmpxchg16 (volatile l4_uint16_t *dest, l4_uint16_t cmp_val, l4_uint16_t new_val)`  
*Atomic compare and exchange (16 bit version)*
- `int l4util_cmpxchg8 (volatile l4_uint8_t *dest, l4_uint8_t cmp_val, l4_uint8_t new_val)`  
*Atomic compare and exchange (8 bit version)*
- `int l4util_cmpxchg (volatile l4_umword_t *dest, l4_umword_t cmp_val, l4_umword_t new_val)`  
*Atomic compare and exchange (machine wide fields)*
- `l4_uint32_t l4util_xchg32 (volatile l4_uint32_t *dest, l4_uint32_t val)`  
*Atomic exchange (32 bit version)*
- `l4_uint16_t l4util_xchg16 (volatile l4_uint16_t *dest, l4_uint16_t val)`  
*Atomic exchange (16 bit version)*
- `l4_uint8_t l4util_xchg8 (volatile l4_uint8_t *dest, l4_uint8_t val)`  
*Atomic exchange (8 bit version)*
- `l4_umword_t l4util_xchg (volatile l4_umword_t *dest, l4_umword_t val)`  
*Atomic exchange (machine wide fields)*
- `void l4util_atomic_add (volatile long *dest, long val)`  
*Atomic add.*
- `void l4util_atomic_inc (volatile long *dest)`  
*Atomic increment.*

## Atomic add/sub/and/or (8,16,32 bit version) without result

- `void l4util_add8 (volatile l4_uint8_t *dest, l4_uint8_t val)`
- `void l4util_add16 (volatile l4_uint16_t *dest, l4_uint16_t val)`
- `void l4util_add32 (volatile l4_uint32_t *dest, l4_uint32_t val)`
- `void l4util_sub8 (volatile l4_uint8_t *dest, l4_uint8_t val)`
- `void l4util_sub16 (volatile l4_uint16_t *dest, l4_uint16_t val)`
- `void l4util_sub32 (volatile l4_uint32_t *dest, l4_uint32_t val)`
- `void l4util_and8 (volatile l4_uint8_t *dest, l4_uint8_t val)`
- `void l4util_and16 (volatile l4_uint16_t *dest, l4_uint16_t val)`
- `void l4util_and32 (volatile l4_uint32_t *dest, l4_uint32_t val)`
- `void l4util_or8 (volatile l4_uint8_t *dest, l4_uint8_t val)`

- void **l4util\_or16** (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- void **l4util\_or32** (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)

#### Atomic add/sub/and/or operations (8,16,32 bit) with result

- [l4\\_uint8\\_t](#) **l4util\_add8\_res** (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t](#) **l4util\_add16\_res** (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t](#) **l4util\_add32\_res** (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t](#) **l4util\_sub8\_res** (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t](#) **l4util\_sub16\_res** (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t](#) **l4util\_sub32\_res** (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t](#) **l4util\_and8\_res** (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t](#) **l4util\_and16\_res** (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t](#) **l4util\_and32\_res** (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t](#) **l4util\_or8\_res** (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t](#) **l4util\_or16\_res** (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t](#) **l4util\_or32\_res** (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)

#### Atomic inc/dec (8,16,32 bit) without result

- void **l4util\_inc8** (volatile [l4\\_uint8\\_t](#) \*dest)
- void **l4util\_inc16** (volatile [l4\\_uint16\\_t](#) \*dest)
- void **l4util\_inc32** (volatile [l4\\_uint32\\_t](#) \*dest)
- void **l4util\_dec8** (volatile [l4\\_uint8\\_t](#) \*dest)
- void **l4util\_dec16** (volatile [l4\\_uint16\\_t](#) \*dest)
- void **l4util\_dec32** (volatile [l4\\_uint32\\_t](#) \*dest)

#### Atomic inc/dec (8,16,32 bit) with result

- [l4\\_uint8\\_t](#) **l4util\_inc8\_res** (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t](#) **l4util\_inc16\_res** (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t](#) **l4util\_inc32\_res** (volatile [l4\\_uint32\\_t](#) \*dest)
- [l4\\_uint8\\_t](#) **l4util\_dec8\_res** (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t](#) **l4util\_dec16\_res** (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t](#) **l4util\_dec32\_res** (volatile [l4\\_uint32\\_t](#) \*dest)

### 15.393.1 Detailed Description

atomic operations header and generic implementations

#### Date

10/20/2000

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de), Jork Loeser [jork@os.inf.tu-dresden.de](mailto:jork@os.inf.tu-dresden.de)

Definition in file [atomic.h](#).

## 15.394 atomic.h

```

00001 /*****
00010 */
00011 * (c) 2000-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 /*****
00019 #ifndef __L4UTIL__INCLUDE__ATOMIC_H__
00020 #define __L4UTIL__INCLUDE__ATOMIC_H__
00021
00022 #include <l4/sys/l4int.h>
00023 #include <l4/sys/compiler.h>
00024
00025 /*****
00026 *** Prototypes
00027 *****/
00028
00029 EXTERN_C_BEGIN
00030
00049 L4_INLINE int
00050 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00051 l4_uint64_t cmp_val, l4_uint64_t new_val);
00052
00066 L4_INLINE int
00067 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00068 l4_uint32_t cmp_val, l4_uint32_t new_val);
00069
00083 L4_INLINE int
00084 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00085 l4_uint16_t cmp_val, l4_uint16_t new_val);
00086
00100 L4_INLINE int
00101 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00102 l4_uint8_t cmp_val, l4_uint8_t new_val);
00103
00117 L4_INLINE int
00118 l4util_cmpxchg(volatile l4_umword_t * dest,
00119 l4_umword_t cmp_val, l4_umword_t new_val);
00120
00130 L4_INLINE l4_uint32_t
00131 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);
00132
00142 L4_INLINE l4_uint16_t
00143 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);
00144
00154 L4_INLINE l4_uint8_t
00155 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);
00156
00166 L4_INLINE l4_umword_t
00167 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);
00168
00170
00176 L4_INLINE void
00177 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
00178 L4_INLINE void
00179 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
00180 L4_INLINE void
00181 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
00182 L4_INLINE void
00183 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
00184 L4_INLINE void
00185 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
00186 L4_INLINE void
00187 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
00188 L4_INLINE void
00189 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
00190 L4_INLINE void
00191 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
00192 L4_INLINE void
00193 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
00194 L4_INLINE void
00195 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
00196 L4_INLINE void
00197 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
00198 L4_INLINE void
00199 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);
00201
00203
00210 L4_INLINE l4_uint8_t
00211 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00212 L4_INLINE l4_uint16_t

```

```

00213 l4util_addl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00214 L4_INLINE l4_uint32_t
00215 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00216 L4_INLINE l4_uint8_t
00217 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00218 L4_INLINE l4_uint16_t
00219 l4util_subl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00220 L4_INLINE l4_uint32_t
00221 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00222 L4_INLINE l4_uint8_t
00223 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00224 L4_INLINE l4_uint16_t
00225 l4util_andl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00226 L4_INLINE l4_uint32_t
00227 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00228 L4_INLINE l4_uint8_t
00229 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00230 L4_INLINE l4_uint16_t
00231 l4util_orl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00232 L4_INLINE l4_uint32_t
00233 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00235
00237
00242 L4_INLINE void
00243 l4util_inc8(volatile l4_uint8_t *dest);
00244 L4_INLINE void
00245 l4util_incl6(volatile l4_uint16_t *dest);
00246 L4_INLINE void
00247 l4util_inc32(volatile l4_uint32_t *dest);
00248 L4_INLINE void
00249 l4util_dec8(volatile l4_uint8_t *dest);
00250 L4_INLINE void
00251 l4util_decl6(volatile l4_uint16_t *dest);
00252 L4_INLINE void
00253 l4util_dec32(volatile l4_uint32_t *dest);
00255
00257
00263 L4_INLINE l4_uint8_t
00264 l4util_inc8_res(volatile l4_uint8_t *dest);
00265 L4_INLINE l4_uint16_t
00266 l4util_incl6_res(volatile l4_uint16_t *dest);
00267 L4_INLINE l4_uint32_t
00268 l4util_inc32_res(volatile l4_uint32_t *dest);
00269 L4_INLINE l4_uint8_t
00270 l4util_dec8_res(volatile l4_uint8_t *dest);
00271 L4_INLINE l4_uint16_t
00272 l4util_decl6_res(volatile l4_uint16_t *dest);
00273 L4_INLINE l4_uint32_t
00274 l4util_dec32_res(volatile l4_uint32_t *dest);
00276
00284 L4_INLINE void
00285 l4util_atomic_add(volatile long *dest, long val);
00286
00293 L4_INLINE void
00294 l4util_atomic_inc(volatile long *dest);
00295
00296 EXTERN_C_END
00297
00298 /*****
00299 * IMPLEMENTATION
00300 *****/
00301
00302 L4_INLINE int
00303 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00304 l4_uint64_t cmp_val, l4_uint64_t new_val)
00305 {
00306 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00307 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00308 }
00309
00310 L4_INLINE int
00311 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00312 l4_uint32_t cmp_val, l4_uint32_t new_val)
00313 {
00314 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00315 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00316 }
00317
00318 L4_INLINE int
00319 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00320 l4_uint16_t cmp_val, l4_uint16_t new_val)
00321 {
00322 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00323 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00324 }
00325
00326 L4_INLINE int

```

```

00327 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00328 l4_uint8_t cmp_val, l4_uint8_t new_val)
00329 {
00330 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00331 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00332 }
00333
00334 L4_INLINE int
00335 l4util_cmpxchg(volatile l4_umword_t * dest,
00336 l4_umword_t cmp_val, l4_umword_t new_val)
00337 {
00338 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00339 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00340 }
00341
00342 L4_INLINE l4_uint32_t
00343 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val)
00344 {
00345 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00346 }
00347
00348 L4_INLINE l4_uint16_t
00349 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val)
00350 {
00351 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00352 }
00353
00354 L4_INLINE l4_uint8_t
00355 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val)
00356 {
00357 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00358 }
00359
00360 L4_INLINE l4_umword_t
00361 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val)
00362 {
00363 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00364 }
00365
00366 L4_INLINE void
00367 l4util_inc8(volatile l4_uint8_t *dest)
00368 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00369
00370 L4_INLINE void
00371 l4util_inc16(volatile l4_uint16_t *dest)
00372 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00373
00374 L4_INLINE void
00375 l4util_inc32(volatile l4_uint32_t *dest)
00376 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00377
00378 L4_INLINE void
00379 l4util_atomic_inc(volatile long *dest)
00380 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00381
00382 L4_INLINE void
00383 l4util_dec8(volatile l4_uint8_t *dest)
00384 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00385
00386 L4_INLINE void
00387 l4util_dec16(volatile l4_uint16_t *dest)
00388 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00389
00390 L4_INLINE void
00391 l4util_dec32(volatile l4_uint32_t *dest)
00392 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00393
00394
00395 L4_INLINE l4_uint8_t
00396 l4util_inc8_res(volatile l4_uint8_t *dest)
00397 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00398
00399 L4_INLINE l4_uint16_t
00400 l4util_inc16_res(volatile l4_uint16_t *dest)
00401 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00402
00403 L4_INLINE l4_uint32_t
00404 l4util_inc32_res(volatile l4_uint32_t *dest)
00405 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00406
00407 L4_INLINE l4_uint8_t
00408 l4util_dec8_res(volatile l4_uint8_t *dest)
00409 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00410
00411 L4_INLINE l4_uint16_t
00412 l4util_dec16_res(volatile l4_uint16_t *dest)
00413 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }

```

```
00414
00415 L4_INLINE l4_uint32_t
00416 l4util_dec32_res(volatile l4_uint32_t *dest)
00417 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00418
00419 L4_INLINE l4_umword_t
00420 l4util_dec_res(volatile l4_umword_t *dest)
00421 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00422
00423 L4_INLINE void
00424 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val)
00425 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00426
00427 L4_INLINE void
00428 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val)
00429 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00430
00431 L4_INLINE void
00432 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val)
00433 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00434
00435 L4_INLINE void
00436 l4util_atomic_add(volatile long *dest, long val)
00437 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00438
00439 L4_INLINE void
00440 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val)
00441 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00442
00443 L4_INLINE void
00444 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val)
00445 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00446
00447 L4_INLINE void
00448 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val)
00449 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00450
00451 L4_INLINE void
00452 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val)
00453 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00454
00455 L4_INLINE void
00456 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val)
00457 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00458
00459 L4_INLINE void
00460 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val)
00461 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00462
00463 L4_INLINE void
00464 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val)
00465 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00466
00467 L4_INLINE void
00468 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val)
00469 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00470
00471 L4_INLINE void
00472 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val)
00473 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00474
00475 L4_INLINE l4_uint8_t
00476 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00477 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00478
00479 L4_INLINE l4_uint16_t
00480 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00481 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00482
00483 L4_INLINE l4_uint32_t
00484 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00485 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00486
00487 L4_INLINE l4_uint8_t
00488 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00489 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00490
00491 L4_INLINE l4_uint16_t
00492 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00493 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00494
00495 L4_INLINE l4_uint32_t
00496 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00497 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00498
00499 L4_INLINE l4_uint8_t
00500 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
```



```

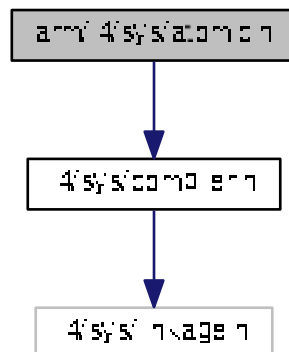
00501 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00502
00503 L4_INLINE l4_uint16_t
00504 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00505 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00506
00507 L4_INLINE l4_uint32_t
00508 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00509 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00510
00511 L4_INLINE l4_uint8_t
00512 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00513 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00514
00515 L4_INLINE l4_uint16_t
00516 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00517 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00518
00519 L4_INLINE l4_uint32_t
00520 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00521 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00522
00523 #endif /* ! __L4UTIL__INCLUDE__ATOMIC_H__ */

```

## 15.395 arm/l4/sys/atomic.h File Reference

Atomic memory modifications.

#include <l4/sys/compiler.h>  
 Include dependency graph for atomic.h:



### 15.395.1 Detailed Description

Atomic memory modifications.

Definition in file [atomic.h](#).

## 15.396 atomic.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027
00028 EXTERN_C long int
00029 l4_atomic_add(volatile long int* mem, long int offset) L4_NOTHROW L4_LONG_CALL;
00030
00031 EXTERN_C long int
00032 l4_atomic_xchg(volatile long int* mem, long int newval) L4_NOTHROW L4_LONG_CALL;
00033
00034 EXTERN_C long int
00035 l4_atomic_cmpxchg(volatile long int* mem, long int oldval, long int newval)
00036 L4_NOTHROW L4_LONG_CALL;

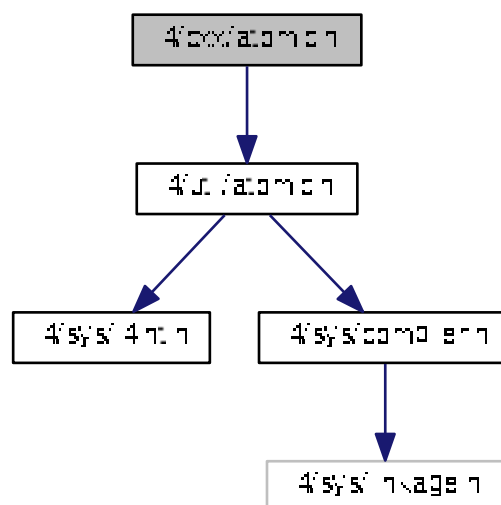
```

## 15.397 l4/cxx/atomic.h File Reference

Atomic template.

```
#include <l4/util/atomic.h>
```

Include dependency graph for atomic.h:



## Namespaces

- [L4](#)

[L4](#) *low-level kernel interface.*

### 15.397.1 Detailed Description

Atomic template.

Definition in file [atomic.h](#).

## 15.398 atomic.h

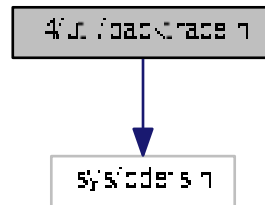
```

00001
00005 /*
00006 * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023 #pragma once
00024
00025 #include <l4/util/atomic.h>
00026
00027 extern "C" void ____error_compare_and_swap_does_not_support_3_bytes____();
00028 extern "C" void ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00029
00030 namespace L4
00031 {
00032 template< typename X >
00033 inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00034 {
00035 switch (sizeof(X))
00036 {
00037 case 1:
00038 return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00039 case 2:
00040 return l4util_cmpxchg16((l4_uint16_t volatile*)dst, old_val, new_val);
00041 case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00042 case 4:
00043 return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00044 default:
00045 ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00046 }
00047 return 0;
00048 }
00049 }
```

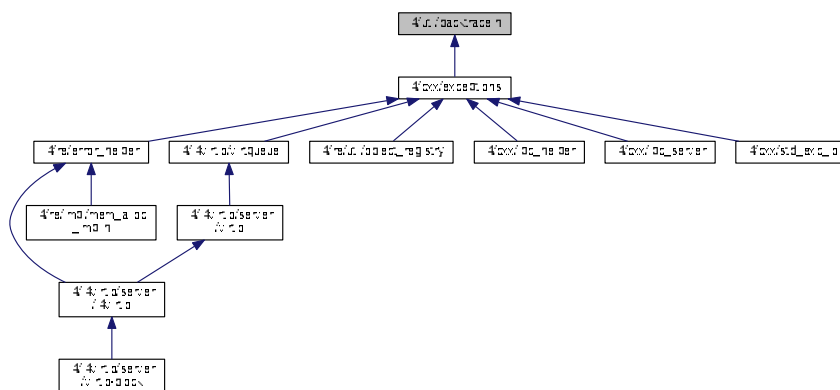
## 15.399 l4/util/backtrace.h File Reference

Backtrace.

```
#include <sys/cdefs.h>
Include dependency graph for backtrace.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- `int l4util_backtrace (void **pc_array, int max_len)`  
*Fill backtrace structure.*

### 15.399.1 Detailed Description

Backtrace.

Definition in file [backtrace.h](#).

## 15.399.2 Function Documentation

## 15.399.2.1 l4util\_backtrace()

```
int l4util_backtrace (
 void ** pc_array,
 int max_len)
```

Fill backtrace structure.

## Parameters

|                 |                                |
|-----------------|--------------------------------|
| <i>pc_array</i> | Array of instruction pointers. |
| <i>max_len</i>  | Length of array.               |

## Returns

Number of entries

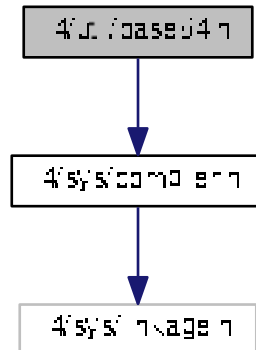
## 15.400 backtrace.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #pragma once
00014
00015 #include <sys/cdefs.h>
00016
00017 __BEGIN_DECLS
00018
00026 int l4util_backtrace(void **pc_array, int max_len);
00027
00028 __END_DECLS
```

## 15.401 l4/util/base64.h File Reference

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

```
#include <lib/sys/compiler.h>
Include dependency graph for base64.h:
```



## Functions

- void [base64\\_encode](#) (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*base-64-encode string infile*
- void [base64\\_decode](#) (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*decode base-64-encoded string infile*

### 15.401.1 Detailed Description

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

#### Date

04/26/2002

#### Author

Joerg Nothnagel [jn6@os.inf.tu-dresden.de](mailto:jn6@os.inf.tu-dresden.de)

Definition in file [base64.h](#).

## 15.402 base64.h

```

00001
00010 /*
00011 * (c) 2008-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 #ifndef B64_EN_DECODE
00019 #define B64_EN_DECODE
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00030
00041 L4_CV void base64_encode(const char *infile, unsigned int in_size, char **outfile);
00042
00053 L4_CV void base64_decode(const char *infile, unsigned int in_size, char **outfile);
00054
00055 EXTERN_C_END
00056
00058 #endif //B64_EN_DECODE

```

## 15.403 l4/util/bitops.h File Reference

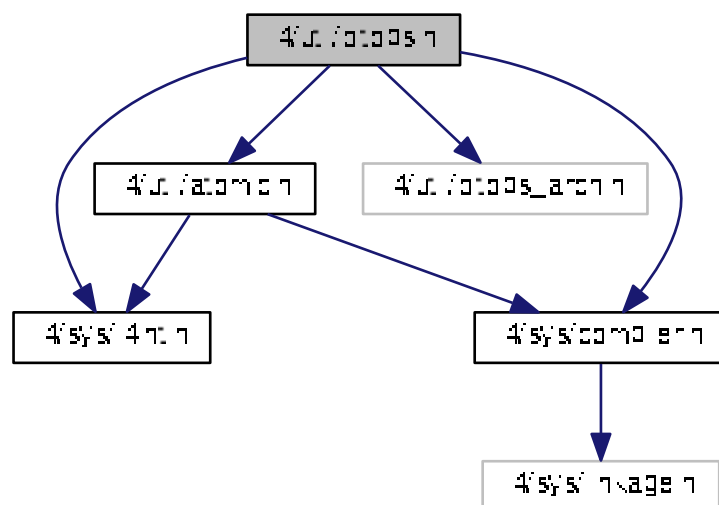
bit manipulation functions

```

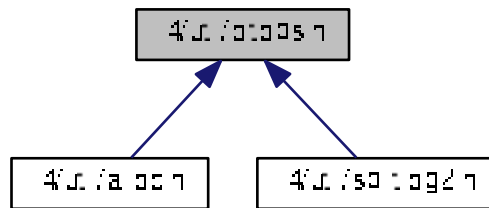
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
#include <l4/util/atomic.h>

```

Include dependency graph for bitops.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define l4util_test_and_clear_bit(b, dest) l4util_btr(b, dest)`  
*define some more usual names*

## Functions

- void `l4util_set_bit` (int b, volatile `l4_umword_t` \*dest)  
*Set bit in memory.*
- void `l4util_clear_bit` (int b, volatile `l4_umword_t` \*dest)  
*Clear bit in memory.*
- void `l4util_complement_bit` (int b, volatile `l4_umword_t` \*dest)  
*Complement bit in memory.*
- int `l4util_test_bit` (int b, const volatile `l4_umword_t` \*dest)  
*Test bit (return value of bit)*
- int `l4util_bts` (int b, volatile `l4_umword_t` \*dest)  
*Bit test and set.*
- int `l4util_btr` (int b, volatile `l4_umword_t` \*dest)  
*Bit test and reset.*
- int `l4util_btc` (int b, volatile `l4_umword_t` \*dest)  
*Bit test and complement.*
- int `l4util_bsr` (`l4_umword_t` word)  
*Bit scan reverse.*
- int `l4util_bsf` (`l4_umword_t` word)  
*Bit scan forward.*
- int `l4util_find_first_set_bit` (const void \*dest, `l4_size_t` size)  
*Find the first set bit in a memory region.*
- int `l4util_find_first_zero_bit` (const void \*dest, `l4_size_t` size)  
*Find the first zero bit in a memory region.*
- int `l4util_next_power2` (const unsigned long val)  
*Find the next power of 2 for a given number.*



### 15.403.1 Detailed Description

bit manipulation functions

#### Date

07/03/2001

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [bitops.h](#).

## 15.404 bitops.h

```

00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #ifndef __L4UTIL__INCLUDE__BITOPS_H__
00019 #define __L4UTIL__INCLUDE__BITOPS_H__
00020
00021 */ L4 includes */
00022 #include <l4/sys/l4int.h>
00023 #include <l4/sys/compiler.h>
00024
00026 #define l4util_test_and_clear_bit(b, dest) l4util_btr(b, dest)
00027 #define l4util_test_and_set_bit(b, dest) l4util_bts(b, dest)
00028 #define l4util_test_and_change_bit(b, dest) l4util_btc(b, dest)
00029 #define l4util_log2(word) l4util_bsr(word)
00030
00031 /*****
00032 *** Prototypes
00033 *****/
00034
00035 EXTERN_C_BEGIN
00036
00049 L4_INLINE void
00050 l4util_set_bit(int b, volatile l4_umword_t * dest);
00051
00059 L4_INLINE void
00060 l4util_clear_bit(int b, volatile l4_umword_t * dest);
00061
00069 L4_INLINE void
00070 l4util_complement_bit(int b, volatile l4_umword_t * dest);
00071
00081 L4_INLINE int
00082 l4util_test_bit(int b, const volatile l4_umword_t * dest);
00083
00095 L4_INLINE int
00096 l4util_bts(int b, volatile l4_umword_t * dest);
00097
00109 L4_INLINE int
00110 l4util_btr(int b, volatile l4_umword_t * dest);
00111
00123 L4_INLINE int
00124 l4util_btc(int b, volatile l4_umword_t * dest);
00125
00137 L4_INLINE int
00138 l4util_bsr(l4_umword_t word);
00139
00151 L4_INLINE int
00152 l4util_bsf(l4_umword_t word);
00153
00164 L4_INLINE int
00165 l4util_find_first_set_bit(const void * dest, l4_size_t size);

```

```

00166
00177 L4_INLINE int
00178 l4util_find_first_zero_bit(const void * dest,
00179 l4_size_t size);
00179
00180
00189 L4_INLINE int
00190 l4util_next_power2(const unsigned long val);
00191
00192 EXTERN_C_END
00193
00194 /*****
00195 *** Implementation of specific version
00196 *****/
00197
00198 #include <l4/util/bitops_arch.h>
00199
00200 /*****
00201 *** Generic implementations
00202 *****/
00203
00204 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00205 #include <l4/util/atomic.h>
00206 L4_INLINE void
00207 l4util_set_bit(int b, volatile l4_umword_t * dest)
00208 {
00209 l4_umword_t oldval, newval;
00210
00211 dest += b / (sizeof(*dest) * 8); /* advance dest to the proper element */
00212 b &= sizeof(*dest) * 8 - 1; /* modulo; cut off all upper bits */
00213
00214 do
00215 {
00216 oldval = *dest;
00217 newval = oldval | (1UL << b);
00218 }
00219 while (!l4util_cmpxchg(dest, oldval, newval));
00220 }
00221 #endif
00222
00223 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00224 #include <l4/util/atomic.h>
00225 L4_INLINE void
00226 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00227 {
00228 l4_umword_t oldval, newval;
00229
00230 dest += b / (sizeof(*dest) * 8);
00231 b &= sizeof(*dest) * 8 - 1;
00232
00233 do
00234 {
00235 oldval = *dest;
00236 newval = oldval & ~(1UL << b);
00237 }
00238 while (!l4util_cmpxchg(dest, oldval, newval));
00239 }
00240 #endif
00241
00242 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00243 L4_INLINE int
00244 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00245 {
00246 dest += b / (sizeof(*dest) * 8);
00247 b &= sizeof(*dest) * 8 - 1;
00248
00249 return (*dest >> b) & 1;
00250 }
00251 #endif
00252
00253 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00254 #include <l4/util/atomic.h>
00255 L4_INLINE int
00256 l4util_bts(int b, volatile l4_umword_t * dest)
00257 {
00258 l4_umword_t oldval, newval;
00259
00260 dest += b / (sizeof(*dest) * 8);
00261 b &= sizeof(*dest) * 8 - 1;
00262
00263 do
00264 {
00265 oldval = *dest;
00266 newval = oldval | (1UL << b);
00267 }
00268 while (!l4util_cmpxchg(dest, oldval, newval));
00269

```

```

00270 /* Return old bit */
00271 return (oldval >> b) & 1;
00272 }
00273 #endif
00274
00275 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00276 #include <l4/util/atomic.h>
00277 L4_INLINE int
00278 l4util_btr(int b, volatile l4_umword_t * dest)
00279 {
00280 l4_umword_t oldval, newval;
00281
00282 dest += b / (sizeof(*dest) * 8);
00283 b &= sizeof(*dest) * 8 - 1;
00284
00285 do
00286 {
00287 oldval = *dest;
00288 newval = oldval & ~(1UL << b);
00289 }
00290 while (!l4util_cmpxchg(dest, oldval, newval));
00291
00292 /* Return old bit */
00293 return (oldval >> b) & 1;
00294 }
00295 #endif
00296
00297 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00298 L4_INLINE int
00299 l4util_bsr(l4_umword_t word)
00300 {
00301 int i;
00302
00303 if (!word)
00304 return -1;
00305
00306 for (i = 8 * sizeof(word) - 1; i >= 0; i--)
00307 if ((1UL << i) & word)
00308 return i;
00309
00310 return -1;
00311 }
00312 #endif
00313
00314 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00315 L4_INLINE int
00316 l4util_bsf(l4_umword_t word)
00317 {
00318 unsigned int i;
00319
00320 if (!word)
00321 return -1;
00322
00323 for (i = 0; i < sizeof(word) * 8; i++)
00324 if ((1UL << i) & word)
00325 return i;
00326
00327 return -1;
00328 }
00329 #endif
00330
00331 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00332 L4_INLINE int
00333 l4util_find_first_zero_bit(const void * dest,
00334 l4_size_t size)
00335 {
00336 l4_size_t i, j;
00337 unsigned long *v = (unsigned long*)dest;
00338
00339 if (!size)
00340 return 0;
00341
00342 size = (size + 31) & ~0x1f; /* Grmbl: adapt to x86 implementation... */
00343
00344 for (i = j = 0; i < size; i++, j++)
00345 {
00346 if (j >= sizeof(*v) * 8)
00347 {
00348 j = 0;
00349 v++;
00350 }
00351 if (!(1UL << j) & *v)
00352 return i;
00353 }
00354 return size + 1;
00355 }
00356 #endif

```

```

00356
00357 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00358 L4_INLINE void
00359 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00360 {
00361 dest += b / (sizeof(*dest) * 8);
00362 b &= sizeof(*dest) * 8 - 1;
00363
00364 *dest ^= 1UL << b;
00365 }
00366 #endif
00367
00368 /*
00369 * Adapted from:
00370 * http://en.wikipedia.org/wiki/Power_of_two#Algorithm_to_find_the_next-highest_power_of_two
00371 */
00372 L4_INLINE int
00373 l4util_next_power2(unsigned long val)
00374 {
00375 unsigned i;
00376
00377 if (val == 0)
00378 return 1;
00379
00380 val--;
00381 for (i=1; i < sizeof(unsigned long)*8; i<=1)
00382 val = val | val >> i;
00383
00384 return val+1;
00385 }
00386
00387
00388 /* Non-implemented version, catch with a linker warning */
00389
00390 extern int __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(void);
00391
00392 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00393 L4_INLINE int
00394 l4util_btc(int b, volatile l4_umword_t * dest)
00395 { (void)b; (void)dest; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return 0; }
00396 #endif
00397
00398 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00399 L4_INLINE int
00400 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00401 { (void)dest; (void)size; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return 0; }
00402 #endif
00403
00404 #endif /* ! __L4UTIL__INCLUDE__BITOPS_H__ */

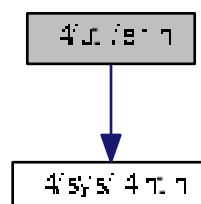
```

## 15.405 l4/util/elf.h File Reference

ELF definition.

```
#include <l4/sys/l4int.h>
```

Include dependency graph for elf.h:



## Data Structures

- struct [Elf32\\_Ehdr](#)  
*ELF32 header.*
- struct [Elf64\\_Ehdr](#)  
*ELF64 header.*
- struct [Elf32\\_Shdr](#)  
*ELF32 section header - figure 1-9, page 1-9.*
- struct [Elf64\\_Shdr](#)  
*ELF64 section header.*
- struct [Elf32\\_Phdr](#)  
*ELF32 program header.*
- struct [Elf64\\_Phdr](#)  
*ELF64 program header.*
- struct [Elf32\\_Dyn](#)  
*ELF32 dynamic entry.*
- struct [Elf64\\_Dyn](#)  
*ELF64 dynamic entry.*
- struct [Elf32\\_Sym](#)  
*ELF32 symbol table entry.*
- struct [Elf64\\_Sym](#)  
*ELF64 symbol table entry.*

## Macros

- #define [EI\\_NIDENT](#) 16  
*number of characters*
- #define [EI\\_CLASS](#) 4  
*ELF class byte index.*
- #define [ELFCLASSNONE](#) 0  
*Invalid ELF class.*
- #define [ELFCLASS32](#) 1  
*32-bit objects*
- #define [ELFCLASS64](#) 2  
*64-bit objects*
- #define [ELFCLASSNUM](#) 3  
*Mask for 32-bit or 64-bit class.*
- #define [EI\\_DATA](#) 5  
*Data encoding byte index.*
- #define [ELFDATANONE](#) 0  
*Invalid data encoding.*
- #define [ELFDATA2LSB](#) 1  
*2's complement, little endian*
- #define [ELFDATA2MSB](#) 2  
*2's complement, big endian*
- #define [EI\\_VERSION](#) 6  
*File version byte index.*
- #define [EI\\_OSABI](#) 7  
*OS ABI identification.*
- #define [ELFOSABI\\_NONE](#) 0

- *UNIX System V ABI.*
- #define [ELFOSABI\\_SYSV](#) 0
- *Alias.*
- #define [ELFOSABI\\_HPUX](#) 1
- *HP-UX.*
- #define [ELFOSABI\\_NETBSD](#) 2
- *NetBSD.*
- #define [ELFOSABI\\_LINUX](#) 3
- *Linux.*
- #define [ELFOSABI\\_SOLARIS](#) 6
- *Sun Solaris.*
- #define [ELFOSABI\\_AIX](#) 7
- *IBM AIX.*
- #define [ELFOSABI\\_IRIX](#) 8
- *SGI Irix.*
- #define [ELFOSABI\\_FREEBSD](#) 9
- *FreeBSD.*
- #define [ELFOSABI\\_TRU64](#) 10
- *Compaq TRU64 UNIX.*
- #define [ELFOSABI\\_MODESTO](#) 11
- *Novell Modesto.*
- #define [ELFOSABI\\_OPENBSD](#) 12
- *OpenBSD.*
- #define [ELFOSABI\\_ARM](#) 97
- *ARM.*
- #define [ELFOSABI\\_STANDALONE](#) 255
- *Standalone (embedded) application.*
- #define [EI\\_ABIVERSION](#) 8
- *ABI version.*
- #define [EI\\_PAD](#) 9
- *Byte index of padding bytes.*
- #define [ET\\_NONE](#) 0
- *no file type*
- #define [ET\\_REL](#) 1
- *relocatable file*
- #define [ET\\_EXEC](#) 2
- *executable file*
- #define [ET\\_DYN](#) 3
- *shared object file*
- #define [ET\\_CORE](#) 4
- *core file*
- #define [ET\\_LOPROC](#) 0xff00
- *processor-specific*
- #define [ET\\_HIPROC](#) 0xffff
- *processor-specific*
- #define [EM\\_NONE](#) 0
- *no machine*
- #define [EM\\_M32](#) 1
- *AT&T WE 32100.*
- #define [EM\\_SPARC](#) 2
- *SPARC.*

- #define [EM\\_386](#) 3  
*Intel 80386.*
- #define [EM\\_68K](#) 4  
*Motorola 68000.*
- #define [EM\\_88K](#) 5  
*Motorola 88000.*
- #define [EM\\_860](#) 7  
*Intel 80860.*
- #define [EM\\_MIPS](#) 8  
*MIPS RS3000 big-endian.*
- #define [EM\\_MIPS\\_RS4\\_BE](#) 10  
*MIPS RS4000 big-endian.*
- #define [EM\\_SPARC64](#) 11  
*SPARC 64-bit.*
- #define [EM\\_PARISC](#) 15  
*HP PA-RISC.*
- #define [EM\\_VPP500](#) 17  
*Fujitsu VPP500.*
- #define [EM\\_SPARC32PLUS](#) 18  
*Sun's V8plus.*
- #define [EM\\_960](#) 19  
*Intel 80960.*
- #define [EM\\_PPC](#) 20  
*PowerPC.*
- #define [EM\\_V800](#) 36  
*NEC V800.*
- #define [EM\\_FR20](#) 37  
*Fujitsu FR20.*
- #define [EM\\_RH32](#) 38  
*TRW RH-32.*
- #define [EM\\_RCE](#) 39  
*Motorola RCE.*
- #define [EM\\_ARM](#) 40  
*Advanced RISC Machines ARM.*
- #define [EM\\_ALPHA](#) 41  
*Digital Alpha.*
- #define [EM\\_SH](#) 42  
*Hitachi SuperH.*
- #define [EM\\_SPARCV9](#) 43  
*SPARC v9 64-bit.*
- #define [EM\\_TRICORE](#) 44  
*Siemens Tricore embedded processor.*
- #define [EM\\_ARC](#) 45  
*Argonaut RISC Core, Argonaut Techn Inc.*
- #define [EM\\_H8\\_300](#) 46  
*Hitachi H8/300.*
- #define [EM\\_H8\\_300H](#) 47  
*Hitachi H8/300H.*
- #define [EM\\_H8S](#) 48  
*Hitachi H8/S.*
- #define [EM\\_H8\\_500](#) 49

- Hitachi H8/500.*
- #define [EM\\_IA\\_64](#) 50
- HP/Intel IA-64.*
- #define [EM\\_MIPS\\_X](#) 51
- Stanford MIPS-X.*
- #define [EM\\_COLDFIRE](#) 52
- Motorola Coldfire.*
- #define [EM\\_68HC12](#) 53
- Motorola M68HC12.*
- #define [EM\\_X86\\_64](#) 62
- Advanced Micro Devices x86-64.*
- #define [EM\\_PDSP](#) 63
- Sony DSP Processor.*
- #define [EM\\_FX66](#) 66
- Siemens FX66 microcontroller.*
- #define [EM\\_ST9PLUS](#) 67
- STMicroelectronics ST9+ 8/16 mc.*
- #define [EM\\_ST7](#) 68
- STmicroelectronics ST7 8 bit mc.*
- #define [EM\\_68HC16](#) 69
- Motorola MC68HC16 microcontroller.*
- #define [EM\\_68HC11](#) 70
- Motorola MC68HC11 microcontroller.*
- #define [EM\\_68HC08](#) 71
- Motorola MC68HC08 microcontroller.*
- #define [EM\\_68HC05](#) 72
- Motorola MC68HC05 microcontroller.*
- #define [EM\\_SVX](#) 73
- Silicon Graphics SVx.*
- #define [EM\\_ST19](#) 74
- STMicroelectronics ST19 8 bit mc.*
- #define [EM\\_VAX](#) 75
- Digital VAX.*
- #define [EM\\_CRIS](#) 76
- Axis Communications 32-bit embedded processor.*
- #define [EM\\_JAVELIN](#) 77
- Infineon Technologies 32-bit embedded processor.*
- #define [EM\\_FIREPATH](#) 78
- Element 14 64-bit DSP Processor.*
- #define [EM\\_ZSP](#) 79
- LSI Logic 16-bit DSP Processor.*
- #define [EM\\_MMIX](#) 80
- Donald Knuth's educational 64-bit processor.*
- #define [EM\\_HUANY](#) 81
- Harvard University machine-independent object files.*
- #define [EM\\_PRISM](#) 82
- SiTera Prism.*
- #define [EM\\_AVR](#) 83
- Atmel AVR 8-bit microcontroller.*
- #define [EM\\_FR30](#) 84
- Fujitsu FR30.*



- #define [EM\\_D10V](#) 85  
*Mitsubishi D10V.*
- #define [EM\\_D30V](#) 86  
*Mitsubishi D30V.*
- #define [EM\\_V850](#) 87  
*NEC v850.*
- #define [EM\\_M32R](#) 88  
*Mitsubishi M32R.*
- #define [EM\\_MN10300](#) 89  
*Matsushita MN10300.*
- #define [EM\\_MN10200](#) 90  
*Matsushita MN10200.*
- #define [EM\\_PJ](#) 91  
*picoJava*
- #define [EM\\_OPENRISC](#) 92  
*OpenRISC 32-bit embedded processor.*
- #define [EM\\_ARC\\_A5](#) 93  
*ARC Cores Tangent-A5.*
- #define [EM\\_XTENSA](#) 94  
*Tensilica Xtensa Architecture.*
- #define [EM\\_ALTERA\\_NIOS2](#) 113  
*Altera Nios II.*
- #define [EM\\_AARCH64](#) 183  
*ARM AARCH64.*
- #define [EM\\_TILEPRO](#) 188  
*Tilera TILEPro.*
- #define [EM\\_MICROBLAZE](#) 189  
*Xilinx MicroBlaze.*
- #define [EM\\_TILEGX](#) 191  
*Tilera TILE-Gx.*
- #define [EV\\_NONE](#) 0  
*Invalid version.*
- #define [EV\\_CURRENT](#) 1  
*Current version.*
- #define [EI\\_MAG0](#) 0  
*file id*
- #define [EI\\_MAG1](#) 1  
*file id*
- #define [EI\\_MAG2](#) 2  
*file id*
- #define [EI\\_MAG3](#) 3  
*file id*
- #define [EI\\_CLASS](#) 4  
*ELF class byte index.*
- #define [EI\\_DATA](#) 5  
*Data encoding byte index.*
- #define [EI\\_VERSION](#) 6  
*File version byte index.*
- #define [EI\\_OSABI](#) 7  
*OS ABI identification.*
- #define [EI\\_ABIVERSION](#) 8

- ABI version.
- #define **EL\_PAD** 9  
Byte index of padding bytes.
- #define **ELFMAG0** 0x7f  
*e\_ident[EL\_MAG0]*
- #define **ELFMAG1** 'E'  
*e\_ident[EL\_MAG1]*
- #define **ELFMAG2** 'L'  
*e\_ident[EL\_MAG2]*
- #define **ELFMAG3** 'F'  
*e\_ident[EL\_MAG3]*
- #define **ELFCLASSNONE** 0  
Invalid ELF class.
- #define **ELFCLASS32** 1  
32-bit object
- #define **ELFCLASS64** 2  
64-bit object
- #define **ELFDATANONE** 0  
Invalid data encoding.
- #define **ELFDATA2LSB** 1  
2's complement, little endian
- #define **ELFDATA2MSB** 2  
2's complement, big endian
- #define **ELFOSABI\_SYSV** 0  
Alias.
- #define **ELFOSABI\_HPUX** 1  
HP-UX.
- #define **ELFOSABI\_STANDALONE** 255  
Standalone (embedded) application.
- #define **SHN\_UNDEF** 0  
undefined section header entry
- #define **SHN\_LORESERVE** 0xff00  
lower bound of reserved indexes
- #define **SHN\_LOPROC** 0xff00  
lower bound of proc spec entr
- #define **SHN\_HIPROC** 0xff1f  
upper bound of proc spec entr
- #define **SHN\_ABS** 0xffff1  
absolute values for ref
- #define **SHN\_COMMON** 0xffff2  
common symbols
- #define **SHN\_HIRESERVE** 0xffff  
upper bound of reserved indexes
- #define **SHT\_INIT\_ARRAY** 14  
Array of constructors.
- #define **SHT\_FINI\_ARRAY** 15  
Array of destructors.
- #define **SHT\_PREINIT\_ARRAY** 16  
Array of pre-constructors.
- #define **SHT\_GROUP** 17  
Section group.

- #define [SHT\\_SYMTAB\\_SHNDX](#) 18  
*Extended section indeces.*
- #define [SHT\\_NUM](#) 19  
*Number of defined types.*
- #define [SHF\\_WRITE](#) 0x1  
*writeable during execution*
- #define [SHF\\_ALLOC](#) 0x2  
*section occupies virt memory*
- #define [SHF\\_EXECINSTR](#) 0x4  
*code section*
- #define [SHF\\_MERGE](#) 0x10  
*Might be merged.*
- #define [SHF\\_STRINGS](#) 0x20  
*Contains nul-terminated strings.*
- #define [SHF\\_INFO\\_LINK](#) 0x40  
*'sh\_info' contains SHT index*
- #define [SHF\\_LINK\\_ORDER](#) 0x80  
*Preserve order after combining.*
- #define [SHF\\_OS\\_NONCONFORMING](#) 0x100  
*Non-standard OS specific handling required.*
- #define [SHF\\_GROUP](#) 0x200  
*Section is member of a group.*
- #define [SHF\\_TLS](#) 0x400  
*Section hold thread-local data.*
- #define [SHF\\_MASKOS](#) 0x0ff00000  
*OS-specific.*
- #define [SHF\\_MASKPROC](#) 0xf0000000  
*proc spec mask*
- #define [PT\\_NULL](#) 0  
*array is unused*
- #define [PT\\_LOAD](#) 1  
*loadable*
- #define [PT\\_DYNAMIC](#) 2  
*dynamic linking information*
- #define [PT\\_INTERP](#) 3  
*path to interpreter*
- #define [PT\\_NOTE](#) 4  
*auxiliary information*
- #define [PT\\_SHLIB](#) 5  
*reserved*
- #define [PT\\_PHDR](#) 6  
*location of the pht itself*
- #define [PT\\_TLS](#) 7  
*Thread-local storage segment.*
- #define [PT\\_NUM](#) 8  
*Number of defined types.*
- #define [PT\\_LOOS](#) 0x60000000  
*os spec.*
- #define [PT\\_HIOS](#) 0x6fffffff  
*os spec.*
- #define [PT\\_LOPROC](#) 0x70000000

- *processor spec.*  
• #define PT\_HIPROC 0x7fffffff
- *processor spec.*  
• #define PT\_GNU\_EH\_FRAME (PT\_LOOS + 0x474e550)
- *EH frame information.*  
• #define PT\_GNU\_STACK (PT\_LOOS + 0x474e551)
- *Flags for stack.*  
• #define PT\_GNU\_RELRO (PT\_LOOS + 0x474e552)
- *Read only after reloc.*  
• #define PT\_L4\_STACK (PT\_LOOS + 0x12)
- *Address of the stack.*  
• #define PT\_L4\_KIP (PT\_LOOS + 0x13)
- *Address of the KIP.*  
• #define PT\_L4\_AUX (PT\_LOOS + 0x14)
- *Address of the AUX structures.*  
• #define NT\_PRSTATUS 1
- *Contains copy of prstatus struct.*  
• #define NT\_FPREGSET 2
- *Contains copy of fpregset struct.*  
• #define NT\_PRPSINFO 3
- *Contains copy of prpsinfo struct.*  
• #define NT\_PRXREG 4
- *Contains copy of prxregset struct.*  
• #define NT\_TASKSTRUCT 4
- *Contains copy of task structure.*  
• #define NT\_PLATFORM 5
- *String from sysinfo(SI\_PLATFORM)*  
• #define NT\_AUXV 6
- *Contains copy of auxv array.*  
• #define NT\_GWINDOWS 7
- *Contains copy of gwindows struct.*  
• #define NT\_ASRS 8
- *Contains copy of asrset struct.*  
• #define NT\_PSTATUS 10
- *Contains copy of pstatus struct.*  
• #define NT\_PSINFO 13
- *Contains copy of psinfo struct.*  
• #define NT\_PRCRED 14
- *Contains copy of prcred struct.*  
• #define NT\_UTSNAME 15
- *Contains copy of utsname struct.*  
• #define NT\_LWPSTATUS 16
- *Contains copy of lwpstatus struct.*  
• #define NT\_LWPSINFO 17
- *Contains copy of lwpinfo struct.*  
• #define NT\_PRFPXREG 20
- *Contains copy of fprxregset struct.*  
• #define NT\_VERSION 1
- *Contains a version string.*  
• #define DT\_NULL 0

*Dynamic Array Tags, d\_tag - figure 2-10, page 2-12.*

- #define [DT\\_NEEDED](#) 1  
*name of a needed library*
- #define [DT\\_PLTRELSZ](#) 2  
*total size of relocation entry*
- #define [DT\\_PLTGOT](#) 3  
*address assoc with prog link table*
- #define [DT\\_HASH](#) 4  
*address of symbol hash table*
- #define [DT\\_STRTAB](#) 5  
*address of string table*
- #define [DT\\_SYMTAB](#) 6  
*address of symbol table*
- #define [DT\\_RELA](#) 7  
*address of relocation table*
- #define [DT\\_RELASZ](#) 8  
*total size of relocation table*
- #define [DT\\_RELAENT](#) 9  
*size of DT\_RELA relocation entry*
- #define [DT\\_STRSZ](#) 10  
*size of the string table*
- #define [DT\\_SYMENT](#) 11  
*size of a symbol table entry*
- #define [DT\\_INIT](#) 12  
*address of initialization function*
- #define [DT\\_FINI](#) 13  
*address of termination function*
- #define [DT\\_SONAME](#) 14  
*name of the shared object*
- #define [DT\\_RPATH](#) 15  
*search library path*
- #define [DT\\_SYMBOLIC](#) 16  
*alter symbol resolution algorithm*
- #define [DT\\_REL](#) 17  
*address of relocation table*
- #define [DT\\_RELSZ](#) 18  
*total size of DT\_REL relocation table*
- #define [DT\\_RELENT](#) 19  
*size of the DT\_REL relocation entry*
- #define [DT\\_PTRREL](#) 20  
*type of relocation entry*
- #define [DT\\_DEBUG](#) 21  
*for debugging purposes*
- #define [DT\\_TEXTREL](#) 22  
*at least on entry changes r/o section*
- #define [DT\\_JMPREL](#) 23  
*address of relocation entries*
- #define [DT\\_BIND\\_NOW](#) 24  
*Process relocations of object.*
- #define [DT\\_INIT\\_ARRAY](#) 25  
*Array with addresses of init fct.*
- #define [DT\\_FINI\\_ARRAY](#) 26

- *Array with addresses of fini fct.*
- #define [DT\\_INIT\\_ARRAYSZ](#) 27  
*Size in bytes of DT\_INIT\_ARRAY.*
- #define [DT\\_FINI\\_ARRAYSZ](#) 28  
*Size in bytes of DT\_FINI\_ARRAY.*
- #define [DT\\_RUNPATH](#) 29  
*Library search path.*
- #define [DT\\_FLAGS](#) 30  
*Flags for the object being loaded.*
- #define [DT\\_ENCODING](#) 32  
*Start of encoded range.*
- #define [DT\\_PREINIT\\_ARRAY](#) 32  
*Array with addresses of preinit fct.*
- #define [DT\\_PREINIT\\_ARRAYSZ](#) 33  
*size in bytes of DT\_PREINIT\_ARRAY*
- #define [DT\\_NUM](#) 34  
*Number used.*
- #define [DT\\_LOOS](#) 0x6000000d  
*Start of OS-specific.*
- #define [DT\\_HIOS](#) 0x6ffff000  
*End of OS-specific.*
- #define [DT\\_LOPROC](#) 0x70000000  
*processor spec.*
- #define [DT\\_HIPROC](#) 0x7ffffff
- #define [DF\\_ORIGIN](#) 0x00000001  
*Object may use DF\_ORIGIN.*
- #define [DF\\_SYMBOLIC](#) 0x00000002  
*Symbol resolutions starts here.*
- #define [DF\\_TEXTREL](#) 0x00000004  
*Object contains text relocations.*
- #define [DF\\_BIND\\_NOW](#) 0x00000008  
*No lazy binding for this object.*
- #define [DF\\_STATIC\\_TLS](#) 0x00000010  
*Module uses the static TLS model.*
- #define [DF\\_1\\_NOW](#) 0x00000001  
*Set RTLD\_NOW for this object.*
- #define [DF\\_1\\_GLOBAL](#) 0x00000002  
*Set RTLD\_GLOBAL for this object.*
- #define [DF\\_1\\_GROUP](#) 0x00000004  
*Set RTLD\_GROUP for this object.*
- #define [DF\\_1\\_NODELETE](#) 0x00000008  
*Set RTLD\_NODELETE for this object.*
- #define [DF\\_1\\_LOADFLTR](#) 0x00000010  
*Trigger filtee loading at runtime.*
- #define [DF\\_1\\_INITFIRST](#) 0x00000020  
*Set RTLD\_INITFIRST for this object.*
- #define [DF\\_1\\_NOOPEN](#) 0x00000040  
*Set RTLD\_NOOPEN for this object.*
- #define [DF\\_1\\_ORIGIN](#) 0x00000080  
*\$ORIGIN must be handled.*

- #define [DF\\_1\\_DIRECT](#) 0x00000100  
*Direct binding enabled.*
- #define [DF\\_1\\_INTERPOSE](#) 0x00000400  
*Object is used to interpose.*
- #define [DF\\_1\\_NODEFLIB](#) 0x00000800  
*Ignore default lib search path.*
- #define [DF\\_1\\_NODUMP](#) 0x00001000  
*Object can't be dldump'ed.*
- #define [DF\\_1\\_CONFALT](#) 0x00002000  
*Configuration alternative created.*
- #define [DF\\_1\\_ENDFILTEE](#) 0x00004000  
*Filtee terminates filters search.*
- #define [DF\\_1\\_DISPRELDNE](#) 0x00008000  
*Disp reloc applied at build time.*
- #define [DF\\_1\\_DISPRELPND](#) 0x00010000  
*Disp reloc applied at run-time.*
- #define [DF\\_P1\\_LAZYLOAD](#) 0x00000001  
*Lazyload following object.*
- #define [DF\\_P1\\_GROUPPERM](#) 0x00000002  
*Symbols from next object are not generally available.*
- #define [R\\_386\\_NONE](#) 0  
*none*
- #define [R\\_386\\_32](#) 1  
*S + A.*
- #define [R\\_386\\_PC32](#) 2  
*S + A - P*
- #define [R\\_386\\_GOT32](#) 3  
*G + A - P.*
- #define [R\\_386\\_PLT32](#) 4  
*L + A - P.*
- #define [R\\_386\\_COPY](#) 5  
*none*
- #define [R\\_386\\_GLOB\\_DAT](#) 6  
*S.*
- #define [R\\_386\\_JMP\\_SLOT](#) 7  
*S.*
- #define [R\\_386\\_RELATIVE](#) 8  
*B + A.*
- #define [R\\_386\\_GOTOFF](#) 9  
*S + A - GOT.*
- #define [R\\_386\\_GOTPC](#) 10  
*GOT + A - P.*
- #define [STB\\_LOCAL](#) 0  
*not visible outside object file*
- #define [STB\\_GLOBAL](#) 1  
*visible to all objects beeing combined*
- #define [STB\\_WEAK](#) 2  
*resemble global symbols*
- #define [STB\\_LOOS](#) 10  
*os specific*
- #define [STB\\_HIOS](#) 12

- os specific*
- #define STB\_LOPROC 13
- proc specific*
- #define STB\_HIPROC 15
- proc specific*
- #define STT\_NOTYPE 0
- symbol's type not specified*
- #define STT\_OBJECT 1
- associated with a data object*
- #define STT\_FUNC 2
- associated with a function or other code*
- #define STT\_SECTION 3
- associated with a section*
- #define STT\_FILE 4
- source file name associated with object*
- #define STT\_LOOS 10
- os specific*
- #define STT\_HIOS 12
- os specific*
- #define STT\_LOPROC 13
- proc specific*
- #define STT\_HIPROC 15
- proc specific*

## Typedefs

### ELF types

- typedef I4\_uint32\_t Elf32\_Addr
- size 4 align 4*
- typedef I4\_uint32\_t Elf32\_Off
- size 4 align 4*
- typedef I4\_uint16\_t Elf32\_Half
- size 2 align 2*
- typedef I4\_uint32\_t Elf32\_Word
- size 4 align 4*
- typedef I4\_int32\_t Elf32\_Sword
- size 4 align 4*
- typedef I4\_uint64\_t Elf64\_Addr
- size 8 align 8*
- typedef I4\_uint64\_t Elf64\_Off
- size 8 align 8*
- typedef I4\_uint16\_t Elf64\_Half
- size 2 align 2*
- typedef I4\_uint32\_t Elf64\_Word
- size 4 align 4*
- typedef I4\_int32\_t Elf64\_Sword
- size 4 align 4*
- typedef I4\_uint64\_t Elf64\_Xword
- size 8 align 8*
- typedef I4\_int64\_t Elf64\_Sxword
- size 8 align 8*



## 15.405.1 Detailed Description

ELF definition.

Date

08/18/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de) Alexander Warg [aw11@os.inf.tu-dresden.de](mailto:aw11@os.inf.tu-dresden.de)

Many structs from "Executable and Linkable Format (ELF)", Portable Formats Specification, Version 1.1 and "↵ System V Application Binary Interface - DRAFT - April 29, 1998" The Santa Cruz Operation, Inc. (see [http↵://www.sco.com/developer/gabi/contents.html](http://www.sco.com/developer/gabi/contents.html))

Definition in file [elf.h](#).

## 15.406 elf.h

```

00001
00019 /*
00020 * (c) 2008-2009 Author(s)
00021 * economic rights: Technische Universität Dresden (Germany)
00022 * This file is part of TUD:OS and distributed under the terms of the
00023 * GNU Lesser General Public License 2.1.
00024 * Please see the COPYING-LGPL-2.1 file for details.
00025 */
00026
00027 /* (c) 2003-2006 Technische Universitaet Dresden
00028 * This file is part of the exec package, which is distributed under
00029 * the terms of the GNU General Public License 2. Please see the
00030 * COPYING file for details. */
00031
00032 #ifndef _L4_EXEC_ELF_H
00033 #define _L4_EXEC_ELF_H
00034
00035 #include <l4/sys/l4int.h>
00036
00047 typedef l4_uint32_t Elf32_Addr;
00048 typedef l4_uint32_t Elf32_Off;
00049 typedef l4_uint16_t Elf32_Half;
00050 typedef l4_uint32_t Elf32_Word;
00051 typedef l4_int32_t Elf32_Sword;
00052 typedef l4_uint64_t Elf64_Addr;
00053 typedef l4_uint64_t Elf64_Off;
00054 typedef l4_uint16_t Elf64_Half;
00055 typedef l4_uint32_t Elf64_Word;
00056 typedef l4_int32_t Elf64_Sword;
00057 typedef l4_uint64_t Elf64_Xword;
00058 typedef l4_int64_t Elf64_Sxword;
00060
00061 #if L4_MWORD_BITS == 64
00062
00066 #define ElfW(type) _ElfW(Elf, 64, type)
00067 #else
00068 #define ElfW(type) _ElfW(Elf, 32, type)
00069 #endif
00070 #define _ElfW(e,w,t) __ElfW(e, w, _##t)
00071 #define __ElfW(e,w,t) e##w##t
00072
00073 #if defined(ARCH_x86)
00074 #define L4_ARCH_EI_DATA ELFDATA2LSB
00075 #define L4_ARCH_E_MACHINE EM_386
00076 #define L4_ARCH_EI_CLASS ELFCLASS32
00077 #elif defined(ARCH_amd64)
00078 #define L4_ARCH_EI_DATA ELFDATA2LSB
00079 #define L4_ARCH_E_MACHINE EM_X86_64
00080 #define L4_ARCH_EI_CLASS ELFCLASS64
00081 #elif defined(ARCH_arm)
00082 #define L4_ARCH_EI_DATA ELFDATA2LSB

```

```

00083 #define L4_ARCH_E_MACHINE EM_ARM
00084 #define L4_ARCH_EI_CLASS ELFCLASS32
00085 #elif defined(ARCH_arm64)
00086 #define L4_ARCH_EI_DATA ELFDATA2LSB
00087 #define L4_ARCH_E_MACHINE EM_AARCH64
00088 #define L4_ARCH_EI_CLASS ELFCLASS64
00089 #elif defined(ARCH_ppc32)
00090 #define L4_ARCH_EI_DATA ELFDATA2MSB
00091 #define L4_ARCH_E_MACHINE EM_PPC
00092 #define L4_ARCH_EI_CLASS ELFCLASS32
00093 #elif defined(ARCH_sparc)
00094 #define L4_ARCH_EI_DATA ELFDATA2MSB
00095 #define L4_ARCH_E_MACHINE EM_SPARC
00096 #define L4_ARCH_EI_CLASS ELFCLASS32
00097 #elif defined(ARCH_mips)
00098 #define L4_ARCH_EI_DATA ELFDATA2LSB
00099 #define L4_ARCH_E_MACHINE EM_MIPS
00100 #ifdef __mips64
00101 #define L4_ARCH_EI_CLASS ELFCLASS64
00102 #else
00103 #define L4_ARCH_EI_CLASS ELFCLASS32
00104 #endif
00105 #else
00106 #warning elf.h: Unsupported build architecture!
00107 #endif
00108
00109
00110 /*****
00111 * ELF Header - figure 1-3, page 1-3 *
00112 *****/
00113
00117
00118 #define EI_NIDENT 16
00122 typedef struct {
00123 unsigned char e_ident[EI_NIDENT];
00124 Elf32_Half e_type;
00125 Elf32_Half e_machine;
00126 Elf32_Word e_version;
00127 Elf32_Addr e_entry;
00128 Elf32_Off e_phoff;
00129 Elf32_Off e_shoff;
00130 Elf32_Word e_flags;
00131 Elf32_Half e_ehsize;
00132 Elf32_Half e_phentsize;
00133 Elf32_Half e_phnum;
00134 Elf32_Half e_shentsize;
00135 Elf32_Half e_shnum;
00136 Elf32_Half e_shstrndx;
00137 } Elf32_Ehdr;
00138
00142 typedef struct {
00143 unsigned char e_ident[EI_NIDENT];
00144 Elf64_Half e_type;
00145 Elf64_Half e_machine;
00146 Elf64_Word e_version;
00147 Elf64_Addr e_entry;
00148 Elf64_Off e_phoff;
00149 Elf64_Off e_shoff;
00150 Elf64_Word e_flags;
00151 Elf64_Half e_ehsize;
00152 Elf64_Half e_phentsize;
00153 Elf64_Half e_phnum;
00154 Elf64_Half e_shentsize;
00155 Elf64_Half e_shnum;
00156 Elf64_Half e_shstrndx;
00157 } Elf64_Ehdr;
00158
00159 #define EI_CLASS 4
00160 #define ELFCLASSNONE 0
00161 #define ELFCLASS32 1
00162 #define ELFCLASS64 2
00163 #define ELFCLASSNUM 3
00165 #define EI_DATA 5
00166 #define ELFDATANONE 0
00167 #define ELFDATA2LSB 1
00168 #define ELFDATA2MSB 2
00169 #define ELFDATANUM 3
00170
00171 #define EI_VERSION 6
00174 #define EI_OSABI 7
00175 #define ELFOSABI_NONE 0
00176 #define ELFOSABI_SYSV 0
00177 #define ELFOSABI_HPUX 1
00178 #define ELFOSABI_NETBSD 2
00179 #define ELFOSABI_LINUX 3
00180 #define ELFOSABI_SOLARIS 6
00181 #define ELFOSABI_AIX 7

```

```
00182 #define ELFOSABI_IRIX 8
00183 #define ELFOSABI_FREEBSD 9
00184 #define ELFOSABI_TRU64 10
00185 #define ELFOSABI_MODESTO 11
00186 #define ELFOSABI_OPENBSD 12
00187 #define ELFOSABI_ARM 97
00188 #define ELFOSABI_STANDALONE 255
00190 #define EI_ABIVERSION 8
00192 #define EI_PAD 9
00194 /* object file type - page 1-3 (e_type) */
00195
00196 #define ET_NONE 0
00197 #define ET_REL 1
00198 #define ET_EXEC 2
00199 #define ET_DYN 3
00200 #define ET_CORE 4
00201 #define ET_LOPROC 0xffff00
00202 #define ET_HIPROC 0xffff
00204 /* required architecture - page 1-4 (e_machine) */
00205
00206 #define EM_NONE 0
00207 #define EM_M32 1
00208 #define EM_SPARC 2
00209 #define EM_386 3
00210 #define EM_68K 4
00211 #define EM_88K 5
00212 #define EM_860 7
00213 #define EM_MIPS 8
00214 #define EM_MIPS_RS4_BE 10
00215 #define EM_SPARC64 11
00216 #define EM_PARISC 15
00217 #define EM_VPP500 17
00218 #define EM_SPARC32PLUS 18
00219 #define EM_960 19
00220 #define EM_PPC 20
00221 #define EM_V800 36
00222 #define EM_FR20 37
00223 #define EM_RH32 38
00224 #define EM_RCE 39
00225 #define EM_ARM 40
00226 #define EM_ALPHA 41
00227 #define EM_SH 42
00228 #define EM_SPARCV9 43
00229 #define EM_TRICORE 44
00230 #define EM_ARC 45
00231 #define EM_H8_300 46
00232 #define EM_H8_300H 47
00233 #define EM_H8S 48
00234 #define EM_H8_500 49
00235 #define EM_IA_64 50
00236 #define EM_MIPS_X 51
00237 #define EM_COLDFIRE 52
00238 #define EM_68HC12 53
00239 #define EM_X86_64 62
00240 #define EM_PDSP 63
00241 #define EM_FX66 66
00242 #define EM_ST9PLUS 67
00243 #define EM_ST7 68
00244 #define EM_68HC16 69
00245 #define EM_68HC11 70
00246 #define EM_68HC08 71
00247 #define EM_68HC05 72
00248 #define EM_SVX 73
00249 #define EM_ST19 74
00250 #define EM_VAX 75
00251 #define EM_CRIS 76
00252 #define EM_JAVELIN 77
00253 #define EM_FIREPATH 78
00254 #define EM_ZSP 79
00255 #define EM_MMIX 80
00256 #define EM_HUANY 81
00257 #define EM_PRISM 82
00258 #define EM_AVR 83
00259 #define EM_FR30 84
00260 #define EM_D10V 85
00261 #define EM_D30V 86
00262 #define EM_V850 87
00263 #define EM_M32R 88
00264 #define EM_MN10300 89
00265 #define EM_MN10200 90
00266 #define EM_PJ 91
00267 #define EM_OPENRISC 92
00268 #define EM_ARC_A5 93
00269 #define EM_XTENSA 94
00270 #define EM_ALTERA_NIOS2 113
00271 #define EM_AARCH64 183
00272 #define EM_TILEPRO 188
```

```

00273 #define EM_MICROBLAZE 189
00274 #define EM_TILEGX 191
00275 #define EM_NUM 192
00276
00277 #if 0
00278 #define EM_ALPHA 0x9026 /* interim value used by Linux until the
00279 committee comes up with a final number */
00280 #define EM_S390 0xA390 /* interim value used for IBM S390 */
00281 #endif
00282
00283 /* object file version - page 1-4 (e_version) */
00284
00285 #define EV_NONE 0
00286 #define EV_CURRENT 1
00288 /* e_ident[] Identification Indexes - figure 1-4, page 1-5 */
00289
00290 #define EI_MAG0 0
00291 #define EI_MAG1 1
00292 #define EI_MAG2 2
00293 #define EI_MAG3 3
00294 #define EI_CLASS 4
00295 #define EI_DATA 5
00296 #define EI_VERSION 6
00297 #define EI_OSABI 7
00298 #define EI_ABIVERSION 8
00299 #define EI_PAD 9
00301 /* magic number - page 1-5 */
00302
00303 #define ELFMAG0 0x7f
00304 #define ELFMAG1 'E'
00305 #define ELFMAG2 'L'
00306 #define ELFMAG3 'F'
00308 /* file class or capacity - page 1-6 */
00309
00310 #define ELFCLASSNONE 0
00311 #define ELFCLASS32 1
00312 #define ELFCLASS64 2
00314 /* data encoding - page 1-6 */
00315
00316 #define ELFDATANONE 0
00317 #define ELFDATA2LSB 1
00318 #define ELFDATA2MSB 2
00320 /* Identify operating system and ABI to which the object is targeted */
00321
00322 #define ELFOSABI_SYSV 0
00323 #define ELFOSABI_HPUX 1
00324 #define ELFOSABI_STANDALONE 255
00327 /* *****/
00328 /* Sections - page 1-8 */
00329 /* *****/
00330
00331 /* special section indexes */
00332
00333 #define SHN_UNDEF 0
00334 #define SHN_LORESERVE 0xff00
00335 #define SHN_LOPROC 0xff00
00336 #define SHN_HIPROC 0xffff
00337 #define SHN_ABS 0xffff1
00338 #define SHN_COMMON 0xffff2
00339 #define SHN_HIRESERVE 0xfffff
00342 typedef struct {
00343 Elf32_Word sh_name;
00344 Elf32_Word sh_type;
00345 Elf32_Word sh_flags;
00346 Elf32_Addr sh_addr;
00347 Elf32_Off sh_offset;
00348 Elf32_Word sh_size;
00349 Elf32_Word sh_link;
00350 Elf32_Word sh_info;
00351 Elf32_Word sh_addralign;
00352 Elf32_Word sh_entsize;
00353 } Elf32_Shdr;
00354
00356 typedef struct {
00357 Elf64_Word sh_name;
00358 Elf64_Word sh_type;
00359 Elf64_Xword sh_flags;
00360 Elf64_Addr sh_addr;
00361 Elf64_Off sh_offset;
00362 Elf64_Xword sh_size;
00363 Elf64_Word sh_link;
00364 Elf64_Word sh_info;
00365 Elf64_Xword sh_addralign;
00366 Elf64_Xword sh_entsize;
00367 } Elf64_Shdr;
00368
00369 /* section type - figure 1-10, page 1-10 */

```

```

00370
00371 #define SHT_NULL 0
00372 #define SHT_PROGBITS 1
00373 #define SHT_SYMTAB 2
00374 #define SHT_STRTAB 3
00375 #define SHT_RELA 4
00376 #define SHT_HASH 5
00377 #define SHT_DYNAMIC 6
00378 #define SHT_NOTE 7
00379 #define SHT_NOBITS 8
00380 #define SHT_REL 9
00381 #define SHT_SHLIB 10
00382 #define SHT_DYNSYM 11
00383 #define SHT_INIT_ARRAY 14
00384 #define SHT_FINI_ARRAY 15
00385 #define SHT_PREINIT_ARRAY 16
00386 #define SHT_GROUP 17
00387 #define SHT_SYMTAB_SHNDX 18
00388 #define SHT_NUM 19
00389 #define SHT_LOOS 0x60000000
00390 #define SHT_HIOS 0x6fffffff
00391 #define SHT_LOPROC 0x70000000
00392 #define SHT_HIPROC 0x7fffffff
00393 #define SHT_LOUSER 0x80000000
00394 #define SHT_HIUSER 0xffffffff
00395
00396 /* section attribute flags - page 1-12, figure 1-12 */
00397
00398 #define SHF_WRITE 0x1
00399 #define SHF_ALLOC 0x2
00400 #define SHF_EXECINSTR 0x4
00401 #define SHF_MERGE 0x10
00402 #define SHF_STRINGS 0x20
00403 #define SHF_INFO_LINK 0x40
00404 #define SHF_LINK_ORDER 0x80
00405 #define SHF_OS_NONCONFORMING 0x100
00407 #define SHF_GROUP 0x200
00408 #define SHF_TLS 0x400
00409 #define SHF_MASKOS 0x0ff00000
00410 #define SHF_MASKPROC 0xf0000000
00413 /*****
00414 */
00415 /*****
00416
00417 typedef struct {
00418 Elf32_Word p_type;
00419 Elf32_Off p_offset;
00420 Elf32_Addr p_vaddr;
00421 Elf32_Addr p_paddr;
00422 Elf32_Word p_filesz;
00423 Elf32_Word p_memsz;
00424 Elf32_Word p_flags;
00425 Elf32_Word p_align;
00426 } Elf32_Phdr;
00427
00428
00429 typedef struct {
00430 Elf64_Word p_type;
00431 Elf64_Word p_flags;
00432 Elf64_Off p_offset;
00433 Elf64_Addr p_vaddr;
00434 Elf64_Addr p_paddr;
00435 Elf64_Xword p_filesz;
00436 Elf64_Xword p_memsz;
00437 Elf64_Xword p_align;
00438 } Elf64_Phdr;
00439
00440
00441 /* segment types - figure 2-2, page 2-3 */
00442
00443 #define PT_NULL 0
00444 #define PT_LOAD 1
00445 #define PT_DYNAMIC 2
00446 #define PT_INTERP 3
00447 #define PT_NOTE 4
00448 #define PT_SHLIB 5
00449 #define PT_PHDR 6
00450 #define PT_TLS 7
00451 #define PT_NUM 8
00452 #define PT_LOOS 0x60000000
00453 #define PT_HIOS 0x6fffffff
00454 #define PT_LOPROC 0x70000000
00455 #define PT_HIPROC 0x7fffffff
00457 #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
00458 #define PT_GNU_STACK (PT_LOOS + 0x474e551)
00459 #define PT_GNU_RELRO (PT_LOOS + 0x474e552)
00461 #define PT_L4_STACK (PT_LOOS + 0x12)
00462 #define PT_L4_KIP (PT_LOOS + 0x13)
00463 #define PT_L4_AUX (PT_LOOS + 0x14)

```

```
00465 /* segment permissions - page 2-3 */
00466
00467 #define PF_X 0x1
00468 #define PF_W 0x2
00469 #define PF_R 0x4
00470 #define PF_MASKOS 0x0ff00000
00471 #define PF_MASKPROC 0x7fffffff
00472
00473 /* Legal values for note segment descriptor types for core files. */
00474
00475 #define NT_PRSTATUS 1
00476 #define NT_FPREGSET 2
00477 #define NT_PRPSINFO 3
00478 #define NT_PRXREG 4
00479 #define NT_TASKSTRUCT 4
00480 #define NT_PLATFORM 5
00481 #define NT_AUXV 6
00482 #define NT_GWINDOWS 7
00483 #define NT_ASRS 8
00484 #define NT_PSTATUS 10
00485 #define NT_PSINFO 13
00486 #define NT_PRCRED 14
00487 #define NT_UTSNAME 15
00488 #define NT_LWPSTATUS 16
00489 #define NT_LWPINFO 17
00490 #define NT_PRFPXREG 20
00492 /* Legal values for the note segment descriptor types for object files. */
00493
00494 #define NT_VERSION 1
00496 /* Dynamic structure - figure 2-9, page 2-12 */
00497
00498 typedef struct {
00500 Elf32_Sword d_tag;
00501 union {
00502 Elf32_Word d_val;
00503 Elf32_Addr d_ptr;
00504 } d_un;
00505 } Elf32_Dyn;
00506
00507 typedef struct {
00509 Elf64_Sxword d_tag;
00510 union {
00511 Elf64_Xword d_val;
00512 Elf64_Addr d_ptr;
00513 } d_un;
00514 } Elf64_Dyn;
00515
00518 #define DT_NULL 0
00519 #define DT_NEEDED 1
00520 #define DT_PLTRELSZ 2
00521 #define DT_PLTGOT 3
00522 #define DT_HASH 4
00523 #define DT_STRTAB 5
00524 #define DT_SYMTAB 6
00525 #define DT_RELA 7
00526 #define DT_RELA_SZ 8
00527 #define DT_RELAENT 9
00528 #define DT_STRSZ 10
00529 #define DT_SYMENT 11
00530 #define DT_INIT 12
00531 #define DT_FINI 13
00532 #define DT_SONAME 14
00533 #define DT_RPATH 15
00534 #define DT_SYMBOLIC 16
00535 #define DT_REL 17
00536 #define DT_RELSZ 18
00537 #define DT_RELENT 19
00538 #define DT_PIRREL 20
00539 #define DT_DEBUG 21
00540 #define DT_TEXTREL 22
00541 #define DT_JMPREL 23
00542 #define DT_BIND_NOW 24
00543 #define DT_INIT_ARRAY 25
00544 #define DT_FINI_ARRAY 26
00545 #define DT_INIT_ARRAYSZ 27
00546 #define DT_FINI_ARRAYSZ 28
00547 #define DT_RUNPATH 29
00548 #define DT_FLAGS 30
00549 #define DT_ENCODING 32
00550 #define DT_PREINIT_ARRAY 32
00551 #define DT_PREINIT_ARRAYSZ 33
00552 #define DT_NUM 34
00553 #define DT_LOOS 0x6000000d
00554 #define DT_HIOS 0x6ffff000
00555 #define DT_LOPROC 0x70000000
00556 #define DT_HIPROC 0x7fffffff
00558 /* Values of 'd_un.d_val' in the DT_FLAGS entry. */
```

```

00559 #define DF_ORIGIN 0x00000001
00560 #define DF_SYMBOLIC 0x00000002
00561 #define DF_TEXTREL 0x00000004
00562 #define DF_BIND_NOW 0x00000008
00563 #define DF_STATIC_TLS 0x00000010
00565 /* State flags selectable in the 'd_un.d_val' element of the DT_FLAGS_1
00566 entry in the dynamic section. */
00567 #define DF_1_NOW 0x00000001
00568 #define DF_1_GLOBAL 0x00000002
00569 #define DF_1_GROUP 0x00000004
00570 #define DF_1_NODELETE 0x00000008
00571 #define DF_1_LOADFLTR 0x00000010
00572 #define DF_1_INITFIRST 0x00000020
00573 #define DF_1_NOOPEN 0x00000040
00574 #define DF_1_ORIGIN 0x00000080
00575 #define DF_1_DIRECT 0x00000100
00576 #define DF_1_TRANS 0x00000200
00577 #define DF_1_INTERPOSE 0x00000400
00578 #define DF_1_NODEFLIB 0x00000800
00579 #define DF_1_NODUMP 0x00001000
00580 #define DF_1_CONFALT 0x00002000
00581 #define DF_1_ENDFILTEE 0x00004000
00582 #define DF_1_DISPRELDNE 0x00008000
00583 #define DF_1_DISPRELPND 0x00010000
00585 /* Flags for the feature selection in DT_FEATURE_1. */
00586 #define DTF_1_PARINIT 0x00000001
00587 #define DTF_1_CONFEXP 0x00000002
00588
00589 /* Flags in the DT_POSFLAG_1 entry effecting only the next DT_* entry. */
00590 #define DF_P1_LAZYLOAD 0x00000001
00591 #define DF_P1_GROUPEXPERM 0x00000002
00594 /* Relocation - page 1-21, figure 1-20 */
00595
00596 typedef struct {
00597 Elf32_Addr r_offset;
00598 Elf32_Word r_info;
00599 } Elf32_Rel;
00600
00601 typedef struct {
00602 Elf32_Addr r_offset;
00603 Elf32_Word r_info;
00604 Elf32_Sword r_addend;
00605 } Elf32_Rela;
00606
00607 typedef struct {
00608 Elf64_Addr r_offset;
00609 Elf64_Xword r_info;
00610 } Elf64_Rel;
00611
00612 typedef struct {
00613 Elf64_Addr r_offset;
00614 Elf64_Xword r_info;
00615 Elf64_Sxword r_addend;
00616 } Elf64_Rela;
00617
00618 #define ELF32_R_SYM(i) ((i)>>8)
00619 #define ELF32_R_TYPE(i) ((unsigned char)(i))
00620 #define ELF32_R_INFO(s,t) (((s)<<8)+(unsigned char)(t))
00621
00622 #define ELF64_R_SYM(i) ((i)>>32)
00623 #define ELF64_R_TYPE(i) ((i)&0xffffffffL)
00624 #define ELF64_R_INFO(s,t) (((s)<<32)+(t)&0xffffffffL)
00625
00626 /* Relocation types (processor specific) - page 1-23, figure 1-22 */
00627
00628 #define R_386_NONE 0
00629 #define R_386_32 1
00630 #define R_386_PC32 2
00631 #define R_386_GOT32 3
00632 #define R_386_PLT32 4
00633 #define R_386_COPY 5
00634 #define R_386_GLOB_DAT 6
00635 #define R_386_JMP_SLOT 7
00636 #define R_386_RELATIVE 8
00637 #define R_386_GOTOFF 9
00638 #define R_386_GOTPC 10
00639 #define R_386_32PLT 11
00640 #define R_386_TLS_TPOFF 14 /* Offset in static TLS block */
00641 #define R_386_TLS_IE 15 /* Address of GOT entry for static TLS
00642 block offset */
00643 #define R_386_TLS_GOTIE 16 /* GOT entry for static TLS block
00644 offset */
00645 #define R_386_TLS_LE 17 /* Offset relative to static TLS
00646 block */
00647 #define R_386_TLS_GD 18 /* Direct 32 bit for GNU version of
00648 general dynamic thread local data */
00649 #define R_386_TLS_LDM 19 /* Direct 32 bit for GNU version of

```

```

00650 local dynamic thread local data
00651 in LE code */
00652 #define R_386_16 20
00653 #define R_386_PC16 21
00654 #define R_386_8 22
00655 #define R_386_PC8 23
00656 #define R_386_TLS_GD_32 24 /* Direct 32 bit for general dynamic
00657 thread local data */
00658 #define R_386_TLS_GD_PUSH 25 /* Tag for pushl in GD TLS code */
00659 #define R_386_TLS_GD_CALL 26 /* Relocation for call to
00660 __tls_get_addr() */
00661 #define R_386_TLS_GD_POP 27 /* Tag for popl in GD TLS code */
00662 #define R_386_TLS_LDM_32 28 /* Direct 32 bit for local dynamic
00663 thread local data in LE code */
00664 #define R_386_TLS_LDM_PUSH 29 /* Tag for pushl in LDM TLS code */
00665 #define R_386_TLS_LDM_CALL 30 /* Relocation for call to
00666 __tls_get_addr() in LDM code */
00667 #define R_386_TLS_LDM_POP 31 /* Tag for popl in LDM TLS code */
00668 #define R_386_TLS_LDO_32 32 /* Offset relative to TLS block */
00669 #define R_386_TLS_IE_32 33 /* GOT entry for negated static TLS
00670 block offset */
00671 #define R_386_TLS_LE_32 34 /* Negated offset relative to static
00672 TLS block */
00673 #define R_386_TLS_DTPMOD32 35 /* ID of module containing symbol */
00674 #define R_386_TLS_DTPOFF32 36 /* Offset in TLS block */
00675 #define R_386_TLS_TPOFF32 37 /* Negated offset in static TLS block */
00676 /* Keep this the last entry. */
00677 #define R_386_NUM 38
00678
00679 /* ARM specific declarations */
00680
00681 /* Processor specific flags for the ELF header e_flags field. */
00682 #define EF_ARM_RELEXEC 0x01
00683 #define EF_ARM_HASENTRY 0x02
00684 #define EF_ARM_INTERWORK 0x04
00685 #define EF_ARM_APCS_26 0x08
00686 #define EF_ARM_APCS_FLOAT 0x10
00687 #define EF_ARM_PIC 0x20
00688 #define EF_ARM_ALIGN8 0x40 /* 8-bit structure alignment is in use */
00689 #define EF_ARM_NEW_ABI 0x80
00690 #define EF_ARM_OLD_ABI 0x100
00691
00692 /* Other constants defined in the ARM ELF spec. version B-01. */
00693 /* NB. These conflict with values defined above. */
00694 #define EF_ARM_SYMSARESORTED 0x04
00695 #define EF_ARM_DYN_SYMS_USE_SEGIDX 0x08
00696 #define EF_ARM_MAPSYMSFIRST 0x10
00697 #define EF_ARM_EABIMASK 0xFF000000
00698
00699 #define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
00700 #define EF_ARM_EABI_UNKNOWN 0x00000000
00701 #define EF_ARM_EABI_VER1 0x01000000
00702 #define EF_ARM_EABI_VER2 0x02000000
00703
00704 /* Additional symbol types for Thumb */
00705 #define STT_ARM_TFUNC 0xd
00706
00707 /* ARM-specific values for sh_flags */
00708 #define SHF_ARM_ENTRYSECT 0x10000000 /* Section contains an entry point */
00709 #define SHF_ARM_COMDEF 0x80000000 /* Section may be multiply defined
00710 in the input to a link step */
00711
00712 /* ARM-specific program header flags */
00713 #define PF_ARM_SB 0x10000000 /* Segment contains the location
00714 addressed by the static base */
00715
00716 /* ARM relocs. */
00717 #define R_ARM_NONE 0 /* No reloc */
00718 #define R_ARM_PC24 1 /* PC relative 26 bit branch */
00719 #define R_ARM_ABS32 2 /* Direct 32 bit */
00720 #define R_ARM_REL32 3 /* PC relative 32 bit */
00721 #define R_ARM_PC13 4
00722 #define R_ARM_ABS16 5 /* Direct 16 bit */
00723 #define R_ARM_ABS12 6 /* Direct 12 bit */
00724 #define R_ARM_THM_ABS5 7
00725 #define R_ARM_ABS8 8 /* Direct 8 bit */
00726 #define R_ARM_SBREL32 9
00727 #define R_ARM_THM_PC22 10
00728 #define R_ARM_THM_PC8 11
00729 #define R_ARM_AMP_VCALL9 12
00730 #define R_ARM_SWI24 13
00731 #define R_ARM_THM_SWI8 14
00732 #define R_ARM_XPC25 15
00733 #define R_ARM_THM_XPC22 16
00734 #define R_ARM_COPY 20 /* Copy symbol at runtime */
00735 #define R_ARM_GLOB_DAT 21 /* Create GOT entry */
00736 #define R_ARM_JUMP_SLOT 22 /* Create PLT entry */

```



```

00737 #define R_ARM_RELATIVE 23 /* Adjust by program base */
00738 #define R_ARM_GOTOFF 24 /* 32 bit offset to GOT */
00739 #define R_ARM_GOTPC 25 /* 32 bit PC relative offset to GOT */
00740 #define R_ARM_GOT32 26 /* 32 bit GOT entry */
00741 #define R_ARM_PLT32 27 /* 32 bit PLT address */
00742 #define R_ARM_ALU_PCREL_7_0 32
00743 #define R_ARM_ALU_PCREL_15_8 33
00744 #define R_ARM_ALU_PCREL_23_15 34
00745 #define R_ARM_LDR_SBREL_11_0 35
00746 #define R_ARM_ALU_SBREL_19_12 36
00747 #define R_ARM_ALU_SBREL_27_20 37
00748 #define R_ARM_GNU_VTENTRY 100
00749 #define R_ARM_GNU_VTINHERIT 101
00750 #define R_ARM_THM_PC11 102 /* thumb unconditional branch */
00751 #define R_ARM_THM_PC9 103 /* thumb conditional branch */
00752 #define R_ARM_RXPC25 249
00753 #define R_ARM_RSBREL32 250
00754 #define R_ARM_THM_RPC22 251
00755 #define R_ARM_RREL32 252
00756 #define R_ARM_RABS22 253
00757 #define R_ARM_RPC24 254
00758 #define R_ARM_RBASE 255
00759 /* Keep this the last entry. */
00760 #define R_ARM_NUM 256
00761
00762 /* AMD x86-64 relocations. */
00763 #define R_X86_64_NONE 0 /* No reloc */
00764 #define R_X86_64_64 1 /* Direct 64 bit */
00765 #define R_X86_64_PC32 2 /* PC relative 32 bit signed */
00766 #define R_X86_64_GOT32 3 /* 32 bit GOT entry */
00767 #define R_X86_64_PLT32 4 /* 32 bit PLT address */
00768 #define R_X86_64_COPY 5 /* Copy symbol at runtime */
00769 #define R_X86_64_GLOB_DAT 6 /* Create GOT entry */
00770 #define R_X86_64_JUMP_SLOT 7 /* Create PLT entry */
00771 #define R_X86_64_RELATIVE 8 /* Adjust by program base */
00772 #define R_X86_64_GOTPCREL 9 /* 32 bit signed PC relative
00773 offset to GOT */
00774 #define R_X86_64_32 10 /* Direct 32 bit zero extended */
00775 #define R_X86_64_32S 11 /* Direct 32 bit sign extended */
00776 #define R_X86_64_16 12 /* Direct 16 bit zero extended */
00777 #define R_X86_64_PC16 13 /* 16 bit sign extended pc relative */
00778 #define R_X86_64_8 14 /* Direct 8 bit sign extended */
00779 #define R_X86_64_PC8 15 /* 8 bit sign extended pc relative */
00780 #define R_X86_64_DTPOFF64 16 /* ID of module containing symbol */
00781 #define R_X86_64_DTPOFF64 17 /* Offset in module's TLS block */
00782 #define R_X86_64_TPOFF64 18 /* Offset in initial TLS block */
00783 #define R_X86_64_TLSGD 19 /* 32 bit signed PC relative offset
00784 to two GOT entries for GD symbol */
00785 #define R_X86_64_TLSLD 20 /* 32 bit signed PC relative offset
00786 to two GOT entries for LD symbol */
00787 #define R_X86_64_DTPOFF32 21 /* Offset in TLS block */
00788 #define R_X86_64_GOTTPOFF 22 /* 32 bit signed PC relative offset
00789 to GOT entry for IE symbol */
00790 #define R_X86_64_TPOFF32 23 /* Offset in initial TLS block */
00791
00792 #define R_X86_64_NUM 24
00793
00794 /* Symbol Table Entry - page 1-17, figure 1-16 */
00795
00796 #define STN_UNDEF 0
00797
00798 typedef struct {
00800 Elf32_Word st_name;
00801 Elf32_Addr st_value;
00802 Elf32_Word st_size;
00803 unsigned char st_info;
00804 unsigned char st_other;
00805 Elf32_Half st_shndx;
00806 } Elf32_Sym;
00807
00808 typedef struct {
00810 Elf64_Word st_name;
00811 unsigned char st_info;
00812 unsigned char st_other;
00813 Elf64_Half st_shndx;
00814 Elf64_Addr st_value;
00815 Elf64_Xword st_size;
00816 } Elf64_Sym;
00817
00818 #define ELF32_ST_BIND(i) ((i)>>4)
00819 #define ELF32_ST_TYPE(i) ((i)&0xf)
00820 #define ELF32_ST_INFO(b,t) (((b)<<4)+((t)&0xf))
00821
00822 #define ELF64_ST_BIND(i) ((i)>>4)
00823 #define ELF64_ST_TYPE(i) ((i)&0xf)
00824 #define ELF64_ST_INFO(b,t) (((b)<<4)+((t)&0xf))
00825

```

```

00826 /* Symbol Binding - page 1-18, figure 1-17 */
00827
00828 #define STB_LOCAL 0
00829 #define STB_GLOBAL 1
00830 #define STB_WEAK 2
00831 #define STB_LOOS 10
00832 #define STB_HIOS 12
00833 #define STB_LOPROC 13
00834 #define STB_HIPROC 15
00836 /* Symbol Types - page 1-19, figure 1-18 */
00837
00838 #define STT_NOTYPE 0
00839 #define STT_OBJECT 1
00840 #define STT_FUNC 2
00841 #define STT_SECTION 3
00842 #define STT_FILE 4
00843 #define STT_LOOS 10
00844 #define STT_HIOS 12
00845 #define STT_LOPROC 13
00846 #define STT_HIPROC 15
00848 enum Elf_ATs
00849 {
00850 AT_NULL = 0,
00851 AT_IGNORE = 1,
00852 AT_EXECD = 2,
00853 AT_PHDR = 3,
00854 AT_PHENT = 4,
00855 AT_PHNUM = 5,
00856 AT_PAGESZ = 6,
00857 AT_BASE = 7,
00858 AT_FLAGS = 8,
00859 AT_ENTRY = 9,
00860 AT_NOTELF = 10,
00861 AT_UID = 11,
00862 AT_EUID = 12,
00863 AT_GID = 13,
00864 AT_EGID = 14,
00865
00866 AT_L4_AUX = 0xf0,
00867 AT_L4_ENV = 0xf1,
00868 };
00869
00870 typedef struct Elf32_Auxv
00871 {
00872 Elf32_Word atype;
00873 Elf32_Word avalue;
00874 } Elf32_Auxv;
00875
00876 typedef struct Elf64_Auxv
00877 {
00878 Elf64_Word atype;
00879 Elf64_Word avalue;
00880 } Elf64_Auxv;
00881
00882 /* Some helpers */
00883 static inline int l4util_elf_check_magic(ElfW(Ehdr) *hdr);
00884 static inline int l4util_elf_check_arch(ElfW(Ehdr) *hdr);
00885 static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) *hdr);
00886
00887
00888 /* Implemeantions */
00889 static inline
00890 int l4util_elf_check_magic(ElfW(Ehdr) *hdr)
00891 {
00892 return hdr->e_ident[EI_MAG0] == ELFMAG0
00893 && hdr->e_ident[EI_MAG1] == ELFMAG1
00894 && hdr->e_ident[EI_MAG2] == ELFMAG2
00895 && hdr->e_ident[EI_MAG3] == ELFMAG3;
00896 }
00897
00898 static inline
00899 int l4util_elf_check_arch(ElfW(Ehdr) *hdr)
00900 {
00901 return hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
00902 && hdr->e_ident[EI_DATA] == L4_ARCH_EI_DATA
00903 && hdr->e_machine == L4_ARCH_E_MACHINE;
00904 }
00905
00906 static inline
00907 ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) *hdr)
00908 {
00909 return (ElfW(Phdr) *) ((char *)hdr + hdr->e_phoff);
00910 }
00913 #endif /* _L4_EXEC_ELF_H */

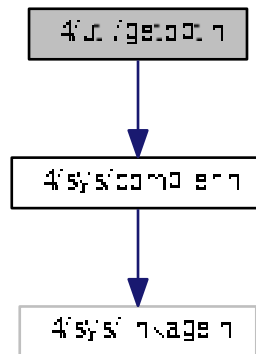
```

## 15.407 l4/util/getopt.h File Reference

getopt

```
#include <l4/sys/compiler.h>
```

Include dependency graph for getopt.h:



### 15.407.1 Detailed Description

getopt

Definition in file [getopt.h](#).

## 15.408 getopt.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef _GETOPT_H
00014 #define _GETOPT_H
00015
00016 #ifndef NULL
00017 #define NULL 0
00018 #endif
00019
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
00023
00024 /* For communication from 'getopt' to the caller.
00025 * When 'getopt' finds an option that takes an argument,
00026 * the argument value is returned here.
00027 * Also, when 'ordering' is RETURN_IN_ORDER,
00028 * each non-option ARGV-element is returned here. */
00029
00030 extern char *optarg;

```

```

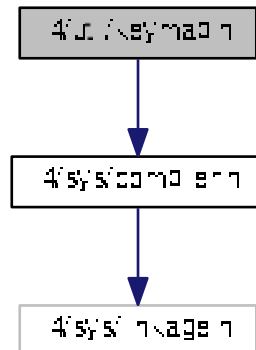
00031
00032 /* Index in ARGV of the next element to be scanned.
00033 This is used for communication to and from the caller
00034 and for communication between successive calls to 'getopt'.
00035
00036 On entry to 'getopt', zero means this is the first call; initialize.
00037
00038 When 'getopt' returns -1, this is the index of the first of the
00039 non-option elements that the caller should itself scan.
00040
00041 Otherwise, 'optind' communicates from one call to the next
00042 how much of ARGV has been scanned so far. */
00043
00044 extern int optind;
00045
00046 /* Callers store zero here to inhibit the error message 'getopt' prints
00047 for unrecognized options. */
00048
00049 extern int opterr;
00050
00051 /* Set to an option character which was unrecognized. */
00052
00053 extern int optopt;
00054
00055 /* Describe the long-named options requested by the application.
00056 The LONG_OPTIONS argument to getopt_long or getopt_long_only is a vector
00057 of 'struct option' terminated by an element containing a name which is
00058 zero.
00059
00060 The field 'has_arg' is:
00061 no_argument (or 0) if the option does not take an argument,
00062 required_argument (or 1) if the option requires an argument,
00063 optional_argument (or 2) if the option takes an optional argument.
00064
00065 If the field 'flag' is not NULL, it points to a variable that is set
00066 to the value given in the field 'val' when the option is found, but
00067 left unchanged if the option is not found.
00068
00069 To have a long-named option do something other than set an 'int' to
00070 a compiled-in constant, such as set a value from 'optarg', set the
00071 option's 'flag' field to zero and its 'val' field to a nonzero
00072 value (the equivalent single-letter option character, if there is
00073 one). For long options that have a zero 'flag' field, 'getopt'
00074 returns the contents of the 'val' field. */
00075
00076 struct option
00077 {
00078 const char *name;
00079 /* has_arg can't be an enum because some compilers complain about
00080 type mismatches in all the code that assumes it is an int. */
00081 int has_arg;
00082 int *flag;
00083 int val;
00084 };
00085
00086 /* Names for the values of the 'has_arg' field of 'struct option'. */
00087
00088 #define no_argument 0
00089 #define required_argument 1
00090 #define optional_argument 2
00091
00092 L4_CV int getopt (int argc, char *const *argv, const char *shortopts);
00093
00094 L4_CV int getopt_long (int argc, char *const *argv, const char *shortopts,
00095 const struct option *longopts, int *longind);
00096 L4_CV int getopt_long_only (int argc, char *const *argv,
00097 const char *shortopts,
00098 const struct option *longopts, int *longind);
00099
00100 L4_CV int _getopt_internal (int argc, char *const *argv,
00101 const char *shortopts,
00102 const struct option *longopts, int *longind,
00103 int long_only);
00104
00105 EXTERN_C_END
00106
00107 #endif /* _GETOPT_H */

```

## 15.409 l4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
Include dependency graph for keymap.h:
```



### 15.409.1 Detailed Description

Event to ASCII key mapping.

Definition in file [keymap.h](#).

## 15.410 keymap.h

```

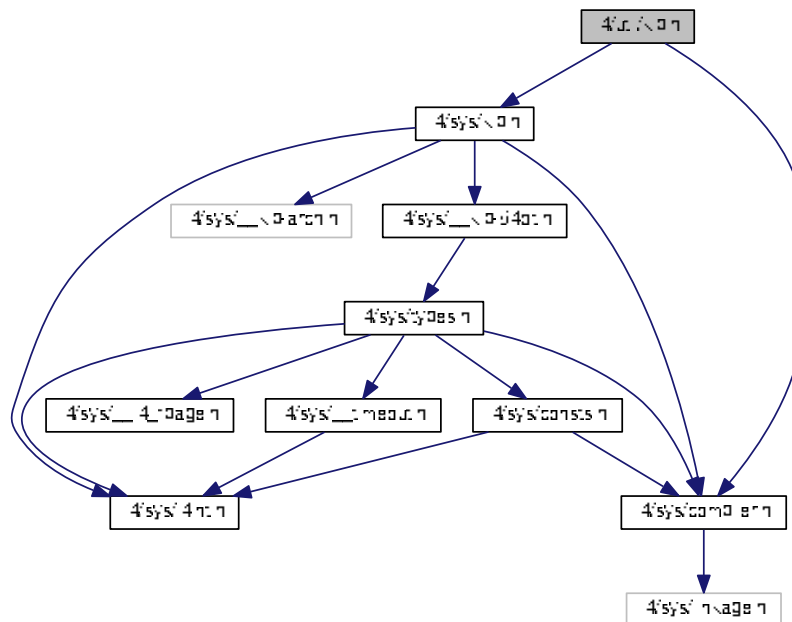
00001
00005 /*
00006 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__KEYMAP_H__
00013 #define __L4UTIL__KEYMAP_H__
00014
00015 #include <l4/sys/compiler.h>
00016
00017 __BEGIN_DECLS
00018
00019 int l4util_map_event_to_keymap(unsigned value, unsigned shift);
00020
00021 __END_DECLS
00022
00023
00024 #endif /* __L4UTIL__KEYMAP_H__ */

```

### 15.411 l4/util/kip.h File Reference

```
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for kip.h:



## Macros

- `#define l4util_kip_for_each_feature(s)` for (`s += strlen(s) + 1; *s; s += strlen(s) + 1`)  
Cycle through kernel features given in the KIP.

## Functions

- `int l4util_kip_kernel_is_ux (l4_kernel_info_t *)`  
Return whether the kernel is running native or under UX.
- `int l4util_kip_kernel_has_feature (l4_kernel_info_t *, const char *str)`  
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t *)`  
Return kernel ABI version.
- `l4_addr_t l4util_memdesc_vm_high (l4_kernel_info_t *kinfo)`  
Return end of virtual memory.

## 15.412 kip.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */

```

```

00013
00014 #pragma once
00015
00016 #include <l4/sys/kip.h>
00017 #include <l4/sys/compiler.h>
00018
00024
00025
00026 EXTERN_C_BEGIN
00027
00036 L4_CV int l4util_kip_kernel_is_ux(l4_kernel_info_t *);
00037
00048 L4_CV int l4util_kip_kernel_has_feature(
 l4_kernel_info_t *, const char *str);
00049
00055 L4_CV unsigned long l4util_kip_kernel_abi_version(
 l4_kernel_info_t *);
00056
00064 L4_CV l4_addr_t l4util_memdesc_vm_high(
 l4_kernel_info_t *kinfo);
00065
00066 EXTERN_C_END
00067
00074 #define l4util_kip_for_each_feature(s) \
00075 for (s += strlen(s) + 1; *s; s += strlen(s) + 1)
00076

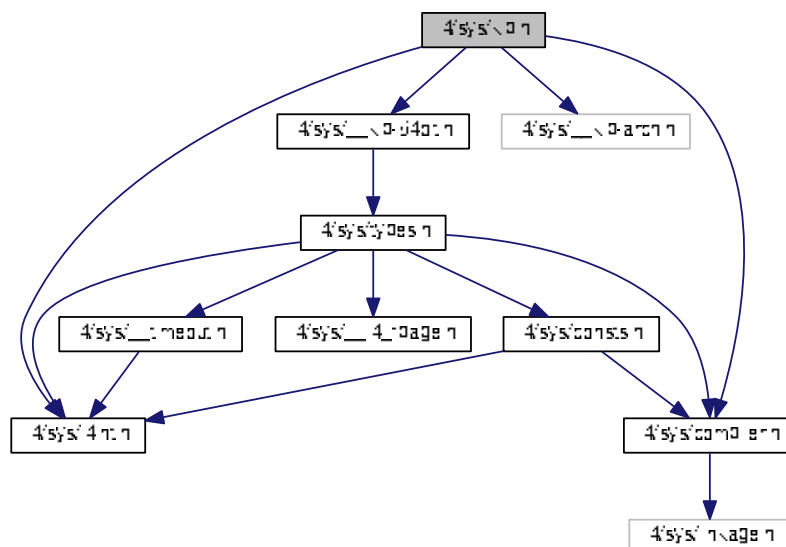
```

## 15.413 I4/sys/kip.h File Reference

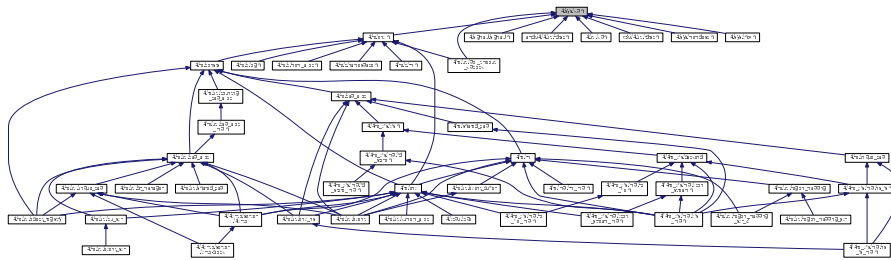
Kernel Info Page access functions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>
#include <l4/sys/__kip-64bit.h>
```

**Include dependency graph for kip.h:**



This graph shows which files directly or indirectly include this file:



## Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`  
Kernel Info Page identifier ("L4μK").

## Functions

- `l4_umword_t l4_kip_version (l4_kernel_info_t *kip) L4_NOTHROW`  
Get the kernel version.
- `const char * l4_kip_version_string (l4_kernel_info_t *kip) L4_NOTHROW`  
Get the kernel version string.
- `int l4_kernel_info_version_offset (l4_kernel_info_t *kip) L4_NOTHROW`  
Return offset in bytes of version\_strings relative to the KIP base.
- `l4_cpu_time_t l4_kip_clock (l4_kernel_info_t *kip) L4_NOTHROW`  
Return clock value from the KIP.
- `l4_umword_t l4_kip_clock_lw (l4_kernel_info_t *kip) L4_NOTHROW`  
Return least significant machine word of clock value from the KIP.

### 15.413.1 Detailed Description

Kernel Info Page access functions.

Definition in file [kip.h](#).

## 15.414 kip.h

```
00001
00006 /*
00007 * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
```



```

00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/l4int.h>
00028
00029 #include <l4/sys/__kip-arch.h>
00030
00034 struct l4_kip_platform_info
00035 {
00036 char name[16];
00037 l4_uint32_t is_mp;
00038 struct l4_kip_platform_info_arch arch;
00039 };
00040
00041 #if L4_MWORD_BITS == 32
00042 # include <l4/sys/__kip-32bit.h>
00043 #else
00044 # include <l4/sys/__kip-64bit.h>
00045 #endif
00046
00054
00058 enum l4_kernel_info_consts_t
00059 {
00060 L4_KIP_VERSION_FIASCO = 0x87004444,
00061 L4_KIP_VERSION_FIASCO_MASK = 0xff00ffff,
00062 };
00063
00067 #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4µK" */
00068
00069
00077 L4_INLINE l4_umword_t l4_kip_version(l4_kernel_info_t *kip)
00078 L4_NOTHROW;
00078
00086 L4_INLINE const char *l4_kip_version_string(
00087 l4_kernel_info_t *kip) L4_NOTHROW;
00087
00096 L4_INLINE int
00097 l4_kernel_info_version_offset(l4_kernel_info_t *kip)
00098 L4_NOTHROW;
00098
00106 L4_INLINE l4_cpu_time_t
00107 l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW;
00108
00116 L4_INLINE l4_umword_t
00117 l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW;
00118
00121 /*****
00122 * Implementations
00123 *****/
00124
00125 L4_INLINE l4_umword_t
00126 l4_kip_version(l4_kernel_info_t *kip) L4_NOTHROW
00127 { return kip->version & L4_KIP_VERSION_FIASCO_MASK; }
00128
00129 L4_INLINE const char*
00130 l4_kip_version_string(l4_kernel_info_t *k)
00131 L4_NOTHROW
00132 { return (const char *)k + l4_kernel_info_version_offset(k); }
00132
00133 L4_INLINE int
00134 l4_kernel_info_version_offset(l4_kernel_info_t *kip)
00135 L4_NOTHROW
00136 { return kip->offset_version_strings << 4; }
00136
00137 L4_INLINE l4_cpu_time_t
00138 l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW
00139 {
00140 unsigned long h1, l;
00141 unsigned long *c;
00142
00143 if (sizeof(unsigned long) == 8)
00144 return kip->_clock_val;
00145
00146 c = (unsigned long *)&kip->_clock_val;
00147 do
00148 {
00149 h1 = c[1];
00150 l4_mb();
00151 l = c[0];
00152 l4_mb();
00153 }
00154 while (h1 != c[1]);
00155
00156 return ((unsigned long long)h1 << 32) | l;

```

```

00157 }
00158
00159 L4_INLINE l4_umword_t
00160 l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW
00161 {
00162 /* We do the casting because the clock field is volatile */
00163 unsigned long *c = (unsigned long *)&kip->_clock_val;
00164 l4_mb();
00165 return c[0];
00166 }

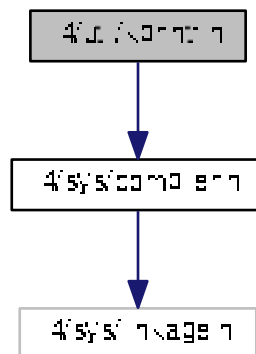
```

## 15.415 l4/util/kprintf.h File Reference

printf using the kernel debugger

```
#include <l4/sys/compiler.h>
```

Include dependency graph for kprintf.h:



### 15.415.1 Detailed Description

printf using the kernel debugger

#### Date

04/05/2007

#### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de),

Definition in file [kprintf.h](#).

## 15.416 kprintf.h

```

00001 /*****/
00009 /*
00010 * (c) 2007-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL__INCLUDE__KPRINTF_H__
00018 #define __L4UTIL__INCLUDE__KPRINTF_H__
00019
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
00023
00024 L4_CV int l4_kprintf(const char *fmt, ...)
00025 __attribute__((format (printf, 1, 2)));
00026
00027 EXTERN_C_END
00028
00029 #endif /* ! __L4UTIL__INCLUDE__KPRINTF_H__ */

```

## 15.417 l4/util/list\_alloc.h File Reference

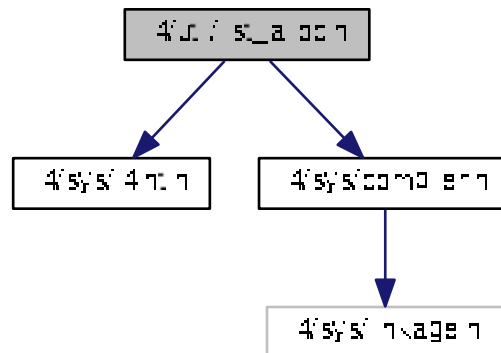
Simple list-based allocator.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for list\_alloc.h:



## Functions

- void `l4la_free` (`l4la_free_t **first`, void `*block`, `l4_size_t` `size`)  
Add free memory to memory pool.
- void `* l4la_alloc` (`l4la_free_t **first`, `l4_size_t` `size`, unsigned `align`)  
Allocate memory from pool.
- void `l4la_dump` (`l4la_free_t **first`)

*Show all list members.*

- void [l4la\\_init](#) ([l4la\\_free\\_t](#) \*\*first)

*Init memory pool.*

- [l4\\_size\\_t](#) [l4la\\_avail](#) ([l4la\\_free\\_t](#) \*\*first)

*Show available memory in pool.*

### 15.417.1 Detailed Description

Simple list-based allocator.

Taken from the Fiasco kernel.

Date

Alexander Warg <aw11os.inf.tu-dresden.de> Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [list\\_alloc.h](#).

### 15.417.2 Function Documentation

#### 15.417.2.1 l4la\_alloc()

```
void* l4la_alloc (
 l4la_free_t ** first,
 l4_size_t size,
 unsigned align)
```

Allocate memory from pool.

Parameters

|              |                                    |
|--------------|------------------------------------|
| <i>first</i> | list identifier                    |
| <i>size</i>  | length of memory block to allocate |
| <i>align</i> | alignment                          |

#### 15.417.2.2 l4la\_avail()

```
l4_size_t l4la_avail (
 l4la_free_t ** first)
```

Show available memory in pool.

## Parameters

|              |                 |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

## 15.417.2.3 l4la\_dump()

```
void l4la_dump (
 l4la_free_t ** first)
```

Show all list members.

## Parameters

|              |                 |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

## 15.417.2.4 l4la\_free()

```
void l4la_free (
 l4la_free_t ** first,
 void * block,
 l4_size_t size)
```

Add free memory to memory pool.

## Parameters

|              |                                |
|--------------|--------------------------------|
| <i>first</i> | list identifier                |
| <i>block</i> | address of unused memory block |
| <i>size</i>  | size of memory block           |

## 15.417.2.5 l4la\_init()

```
void l4la_init (
 l4la_free_t ** first)
```

Init memory pool.

## Parameters

|              |                 |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

## 15.418 list\_alloc.h

```

00001
00008 /*
00009 * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00010 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef L4UTIL_L4LA_H
00018 #define L4UTIL_L4LA_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 typedef struct l4la_free_t_s
00024 {
00025 struct l4la_free_t_s *next;
00026 l4_size_t size;
00027 } l4la_free_t;
00028
00029 #define L4LA_INITIALIZER { 0 }
00030
00031 EXTERN_C_BEGIN
00032
00037 L4_CV void l4la_free(l4la_free_t **first, void *block,
00038 l4_size_t size);
00038
00043 L4_CV void* l4la_alloc(l4la_free_t **first, l4_size_t size, unsigned align);
00044
00047 L4_CV void l4la_dump(l4la_free_t **first);
00048
00051 L4_CV void l4la_init(l4la_free_t **first);
00052
00055 L4_CV l4_size_t l4la_avail(l4la_free_t **first);
00056
00057 EXTERN_C_END
00058
00059 #endif

```

## 15.419 l4/util/lock.h File Reference

Simple lock implementation.

```

#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>

```



```

00026
00027 typedef l4_uint32_t l4util_simple_lock_t;
00028
00029 L4_INLINE int l4_simple_try_lock(l4util_simple_lock_t *lock);
00030 L4_INLINE void l4_simple_unlock(l4util_simple_lock_t *lock);
00031 L4_INLINE int l4_simple_lock_locked(l4util_simple_lock_t *lock);
00032 L4_INLINE void l4_simple_lock_solid(register l4util_simple_lock_t *p);
00033 L4_INLINE void l4_simple_lock(l4util_simple_lock_t * lock);
00034
00035 L4_INLINE int
00036 l4_simple_try_lock(l4util_simple_lock_t *lock)
00037 {
00038 return l4util_xchg32(lock, 1) == 0;
00039 }
00040
00041 L4_INLINE void
00042 l4_simple_unlock(l4util_simple_lock_t *lock)
00043 {
00044 *lock = 0;
00045 }
00046
00047 L4_INLINE int
00048 l4_simple_lock_locked(l4util_simple_lock_t *lock)
00049 {
00050 return (*lock == 0) ? 0 : 1;
00051 }
00052
00053 L4_INLINE void
00054 l4_simple_lock_solid(register l4util_simple_lock_t *p)
00055 {
00056 while (l4_simple_lock_locked(p) || !l4_simple_try_lock(p))
00057 l4_thread_switch(L4_INVALID_CAP);
00058 }
00059
00060 L4_INLINE void
00061 l4_simple_lock(l4util_simple_lock_t * lock)
00062 {
00063 if (!l4_simple_try_lock(lock))
00064 l4_simple_lock_solid(lock);
00065 }
00066
00067 EXTERN_C_END
00068
00069 #endif

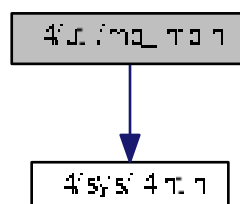
```

## 15.421 l4/util/mb\_info.h File Reference

Multiboot info structure as defined by GRUB.

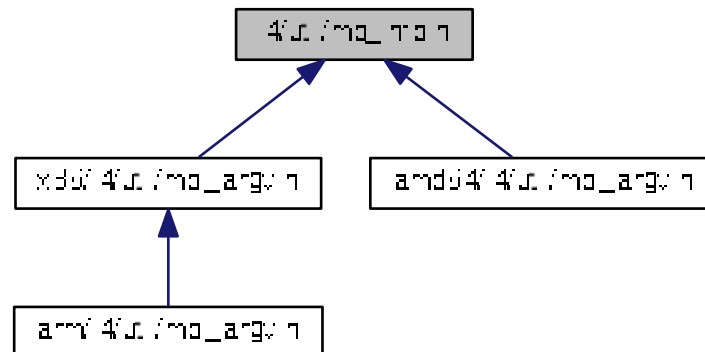
```
#include <l4/sys/l4int.h>
```

Include dependency graph for mb\_info.h:





This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4util\\_mb\\_mod\\_t](#)
- struct [l4util\\_mb\\_addr\\_range\\_t](#)  
*INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.*
- struct [l4util\\_mb\\_drive\\_t](#)  
*Drive Info structure.*
- struct [l4util\\_mb\\_apm\\_t](#)  
*APM BIOS info.*
- struct [l4util\\_mb\\_vbe\\_ctrl\\_t](#)  
*VBE controller information.*
- struct [l4util\\_mb\\_vbe\\_mode\\_t](#)  
*VBE mode information.*
- struct [l4util\\_mb\\_info\\_t](#)

## Macros

- `#define MB_ARD_MEMORY 1`  
*usable memory "Type", all others are reserved.*
- `#define MB_ART_MEMORY 1`  
*Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.*
- `#define MB_ART_RESERVED 2`  
*in use or reserved by system*
- `#define MB_ART_ACPI 3`  
*ACPI Reclaim Memory (RAM that contains ACPI tables)*
- `#define MB_ART_NVS 4`  
*ACPI NVS Memory (must not be used by the OS.*
- `#define MB_ART_UNUSABLE 5`  
*memory in which errors have been detected*
- `#define L4UTIL_MB_MEMORY 0x00000001`

- Flags to be set in the 'flags' parameter above.*
- `#define L4UTIL_MB_BOOTDEV 0x00000002`  
*is there a boot device set?*
  - `#define L4UTIL_MB_CMDLINE 0x00000004`  
*is the command-line defined?*
  - `#define L4UTIL_MB_MODS 0x00000008`  
*are there modules to do something with?*
  - `#define L4UTIL_MB_AOUT_SYMS 0x00000010`  
*is there a symbol table loaded?*
  - `#define L4UTIL_MB_ELF_SHDR 0x00000020`  
*is there an ELF section header table?*
  - `#define L4UTIL_MB_MEM_MAP 0x00000040`  
*is there a full memory map?*
  - `#define L4UTIL_MB_DRIVE_INFO 0x00000080`  
*Is there drive info?*
  - `#define L4UTIL_MB_CONFIG_TABLE 0x00000100`  
*Is there a config table?*
  - `#define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200`  
*Is there a boot loader name?*
  - `#define L4UTIL_MB_APM_TABLE 0x00000400`  
*Is there a APM table?*
  - `#define L4UTIL_MB_VIDEO_INFO 0x00000800`  
*Is there video information?*
  - `#define L4UTIL_MB_VALID 0x2BADB002UL`  
*If we are multiboot-compliant, this value is present in the eax register.*

### 15.421.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file [mb\\_info.h](#).

### 15.421.2 Macro Definition Documentation

#### 15.421.2.1 L4UTIL\_MB\_MEMORY

```
#define L4UTIL_MB_MEMORY 0x00000001
```

Flags to be set in the 'flags' parameter above.

is there basic lower/upper memory information?

Definition at line 263 of file [mb\\_info.h](#).

## 15.421.2.2 MB\_ARD\_MEMORY

```
#define MB_ARD_MEMORY 1
```

usable memory "Type", all others are reserved.

Definition at line 62 of file [mb\\_info.h](#).

## 15.421.2.3 MB\_ART\_MEMORY

```
#define MB_ART_MEMORY 1
```

Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.

390). Other values are undefined.available, usable RAM

Definition at line 68 of file [mb\\_info.h](#).

## 15.422 mb\_info.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013
00014 #ifndef L4UTIL_MB_INFO_H
00015 #define L4UTIL_MB_INFO_H
00016
00017 /*****
00018 * Multiboot (v1)
00019 *****/
00020
00021 #ifndef __ASSEMBLY__
00022
00023 #include <l4/sys/l4int.h>
00024
00031 typedef struct
00032 {
00033 l4_uint32_t mod_start;
00034 l4_uint32_t mod_end;
00035 l4_uint32_t cmdline;
00036 l4_uint32_t pad;
00037 } l4util_mb_mod_t;
00038
00039
00047 typedef struct __attribute__((packed))
00048 {
00049 l4_uint32_t struct_size;
00050 l4_uint64_t addr;
00051 l4_uint64_t size;
00052 l4_uint32_t type;
00053 /* unspecified optional padding... */
00054 } l4util_mb_addr_range_t;
00055
00056 #define l4util_mb_for_each_mmap_entry(i, mbi) \
00057 for (i = (l4util_mb_addr_range_t *) (unsigned long) mbi->mmap_addr; \
00058 (unsigned long) i < (unsigned long) mbi->mmap_addr + mbi->mmap_length; \
00059 i = (l4util_mb_addr_range_t *) ((unsigned long) i + mmap->struct_size + sizeof (mmap->struct_size)))
00060
00062 #define MB_ARD_MEMORY 1
00063
00068 #define MB_ART_MEMORY 1
00069 #define MB_ART_RESERVED 2
```

```
00070 #define MB_ART ACPI 3
00072 #define MB_ART_NV 4
00073 #define MB_ART_UNUSABLE 5
00077 typedef struct
00078 {
00079 l4_uint32_t size;
00080 l4_uint8_t drive_number;
00081 l4_uint8_t drive_mode;
00082 l4_uint16_t drive_cylinders;
00083 l4_uint8_t drive_heads;
00084 l4_uint8_t drive_sectors;
00085 l4_uint16_t drive_ports[0];
00086 } l4util_mb_drive_t;
00087
00088 /* Drive Mode. */
00089 #define MB_DI_CHS_MODE 0
00090 #define MB_DI_LBA_MODE 1
00091
00092
00094 typedef struct
00095 {
00096 l4_uint16_t version;
00097 l4_uint16_t cseg;
00098 l4_uint32_t offset;
00099 l4_uint16_t cseg_l6;
00100 l4_uint16_t dseg_l6;
00101 l4_uint16_t cseg_len;
00102 l4_uint16_t cseg_l6_len;
00103 l4_uint16_t dseg_l6_len;
00104 } l4util_mb_apm_t;
00105
00106
00108 typedef struct
00109 {
00110 l4_uint8_t signature[4];
00111 l4_uint16_t version;
00112 l4_uint32_t oem_string;
00113 l4_uint32_t capabilities;
00114 l4_uint32_t video_mode;
00115 l4_uint16_t total_memory;
00116 l4_uint16_t oem_software_rev;
00117 l4_uint32_t oem_vendor_name;
00118 l4_uint32_t oem_product_name;
00119 l4_uint32_t oem_product_rev;
00120 l4_uint8_t reserved[222];
00121 l4_uint8_t oem_data[256];
00122 } __attribute__((packed)) l4util_mb_vbe_ctrl_t;
00123
00124
00126 typedef struct
00127 {
00130 l4_uint16_t mode_attributes;
00131 l4_uint8_t win_a_attributes;
00132 l4_uint8_t win_b_attributes;
00133 l4_uint16_t win_granularity;
00134 l4_uint16_t win_size;
00135 l4_uint16_t win_a_segment;
00136 l4_uint16_t win_b_segment;
00137 l4_uint32_t win_func;
00138 l4_uint16_t bytes_per_scanline;
00143 l4_uint16_t x_resolution;
00144 l4_uint16_t y_resolution;
00145 l4_uint8_t x_char_size;
00146 l4_uint8_t y_char_size;
00147 l4_uint8_t number_of_planes;
00148 l4_uint8_t bits_per_pixel;
00149 l4_uint8_t number_of_banks;
00150 l4_uint8_t memory_model;
00151 l4_uint8_t bank_size;
00152 l4_uint8_t number_of_image_pages;
00153 l4_uint8_t reserved0;
00158 l4_uint8_t red_mask_size;
00159 l4_uint8_t red_field_position;
00160 l4_uint8_t green_mask_size;
00161 l4_uint8_t green_field_position;
00162 l4_uint8_t blue_mask_size;
00163 l4_uint8_t blue_field_position;
00164 l4_uint8_t reserved_mask_size;
00165 l4_uint8_t reserved_field_position;
00166 l4_uint8_t direct_color_mode_info;
00171 l4_uint32_t phys_base;
00172 l4_uint32_t reserved1;
00173 l4_uint16_t reversed2;
00178 l4_uint16_t linear_bytes_per_scanline;
00179 l4_uint8_t banked_number_of_image_pages;
00180 l4_uint8_t linear_number_of_image_pages;
00181 l4_uint8_t linear_red_mask_size;
```

```

00182 14_uint8_t linear_red_field_position;
00183 14_uint8_t linear_green_mask_size;
00184 14_uint8_t linear_green_field_position;
00185 14_uint8_t linear_blue_mask_size;
00186 14_uint8_t linear_blue_field_position;
00187 14_uint8_t linear_reserved_mask_size;
00188 14_uint8_t linear_reserved_field_position;
00189 14_uint32_t max_pixel_clock;
00190
00191 /* The VBE spec says this structure should have a size of 256 bytes but
00192 * the described structure layout is only 255 bytes... */
00193 14_uint8_t reserved3[189 + 1];
00195 } __attribute__((packed)) 14util_mb_vbe_mode_t;
00196
00197
00206 typedef struct
00207 {
00208 14_uint32_t flags;
00209 14_uint32_t mem_lower;
00210 14_uint32_t mem_upper;
00211 14_uint32_t boot_device;
00212 14_uint32_t cmdline;
00213 14_uint32_t mods_count;
00214 14_uint32_t mods_addr;
00216 union
00217 {
00218 struct
00219 {
00221 14_uint32_t tabsize;
00222 14_uint32_t strsize;
00223 14_uint32_t addr;
00224 14_uint32_t pad;
00225 }
00226 a;
00227
00228 struct
00229 {
00231 14_uint32_t num;
00232 14_uint32_t size;
00233 14_uint32_t addr;
00234 14_uint32_t shndx;
00235 }
00236 e;
00237 }
00238 syms;
00239
00240 14_uint32_t mmap_length;
00241 14_uint32_t mmap_addr;
00242 14_uint32_t drives_length;
00243 14_uint32_t drives_addr;
00244 14_uint32_t config_table;
00245 14_uint32_t boot_loader_name;
00246 14_uint32_t apm_table;
00247 14_uint32_t vbe_ctrl_info;
00248 14_uint32_t vbe_mode_info;
00249 14_uint16_t vbe_mode;
00250 14_uint16_t vbe_interface_seg;
00251 14_uint16_t vbe_interface_off;
00252 14_uint16_t vbe_interface_len;
00253 } 14util_mb_info_t;
00254
00255 #endif /* ! __ASSEMBLY__ */
00256
00262 #define L4UTIL_MB_MEMORY 0x00000001
00263
00265 #define L4UTIL_MB_BOOTDEV 0x00000002
00266
00268 #define L4UTIL_MB_CMDLINE 0x00000004
00269
00271 #define L4UTIL_MB_MODS 0x00000008
00272
00273 /* These next two are mutually exclusive */
00275 #define L4UTIL_MB_AOUT_SYMS 0x00000010
00276
00278 #define L4UTIL_MB_ELF_SHDR 0x00000020
00279
00281 #define L4UTIL_MB_MEM_MAP 0x00000040
00282
00284 #define L4UTIL_MB_DRIVE_INFO 0x00000080
00285
00287 #define L4UTIL_MB_CONFIG_TABLE 0x00000100
00288
00290 #define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200
00291
00293 #define L4UTIL_MB_APM_TABLE 0x00000400
00294
00296 #define L4UTIL_MB_VIDEO_INFO 0x00000800

```

```

00297
00298
00300 #define L4UTIL_MB_VALID 0x2BADB002UL
00301 #define L4UTIL_MB_VALID_ASM 0x2BADB002
00302
00303
00304 /*****
00305 * Multiboot2
00306 *****/
00307
00308 #ifndef __ASSEMBLY__
00309
00310 typedef struct
00311 {
00312 l4_uint32_t total_size;
00313 l4_uint32_t reserved;
00314 } __attribute__((packed)) l4util_mb2_info_t;
00315
00316 typedef struct
00317 {
00318 char string[0];
00319 } __attribute__((packed)) l4util_mb2_cmdline_tag_t;
00320
00321 typedef struct
00322 {
00323 l4_uint32_t mod_start;
00324 l4_uint32_t mod_end;
00325 char string[];
00326 } __attribute__((packed)) l4util_mb2_module_tag_t;
00327
00328 typedef struct
00329 {
00330 l4_uint64_t base_addr;
00331 l4_uint64_t length;
00332 l4_uint32_t type;
00333 l4_uint32_t reserved;
00334 } __attribute__((packed)) l4util_mb2_memmap_entry_t;
00335
00336 typedef struct
00337 {
00338 l4_uint32_t entry_size;
00339 l4_uint32_t entry_version;
00340 l4util_mb2_memmap_entry_t entries[];
00341 } __attribute__((packed)) l4util_mb2_memmap_tag_t;
00342
00343 typedef struct
00344 {
00345 char data[0];
00346 } __attribute__((packed)) l4util_mb2_rsdp_tag_t;
00347
00348 typedef struct
00349 {
00350 l4_uint32_t type;
00351 l4_uint32_t size;
00352
00353 union
00354 {
00355 l4util_mb2_cmdline_tag_t cmdline;
00356 l4util_mb2_module_tag_t module;
00357 l4util_mb2_memmap_tag_t memmap;
00358 l4util_mb2_rsdp_tag_t rsdp;
00359 };
00360 } __attribute__((packed)) l4util_mb2_tag_t;
00361
00362 #endif /* ! __ASSEMBLY__ */
00363
00364
00365 #define L4UTIL_MB2_MAGIC 0xE85250D6
00366 #define L4UTIL_MB2_ARCH_I386 0x0
00367
00368 #define L4UTIL_MB2_TERMINATOR_HEADER_TAG 0
00369 #define L4UTIL_MB2_INFO_REQUEST_HEADER_TAG 1
00370 #define L4UTIL_MB2_ENTRY_ADDRESS_HEADER_TAG 3
00371
00372 #define L4UTIL_MB2_TAG_FLAG_REQUIRED 0
00373
00374 #define L4UTIL_MB2_TAG_ALIGN_SHIFT 3
00375 #define L4UTIL_MB2_TAG_ALIGN 8
00376
00377 #define L4UTIL_MB2_TERMINATOR_INFO_TAG 0
00378 #define L4UTIL_MB2_BOOT_CMDLINE_INFO_TAG 1
00379 #define L4UTIL_MB2_MODULE_INFO_TAG 3
00380 #define L4UTIL_MB2_MEMORY_MAP_INFO_TAG 6
00381 #define L4UTIL_MB2_RSDP_OLD_INFO_TAG 14
00382 #define L4UTIL_MB2_RSDP_NEW_INFO_TAG 15
00383
00384 #endif

```

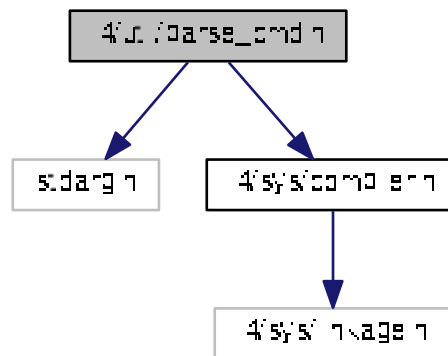
## 15.423 l4/util/parse\_cmd.h File Reference

comfortable command-line parsing

```
#include <stdarg.h>
```

```
#include <l4/sys/compiler.h>
```

Include dependency graph for parse\_cmd.h:



### Typedefs

- typedef void(\* [parse\\_cmd\\_fn\\_t](#)) (int)  
*Function type for PARSE\_CMD\_FN.*
- typedef void(\* [parse\\_cmd\\_fn\\_arg\\_t](#)) (int, const char \*, int)  
*Function type for PARSE\_CMD\_FN\_ARG.*

### Enumerations

- enum [parse\\_cmd\\_type](#)  
*Types for parsing.*

### Functions

- int [parse\\_cmdline](#) (int \*argc, const char \*\*\*argv, char arg0,...)  
*Parse the command-line for specified arguments and store the values into variables.*

#### 15.423.1 Detailed Description

comfortable command-line parsing

Date

2002

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [parse\\_cmd.h](#).

## 15.424 parse\_cmd.h

```

00001
00009 /*
00010 * (c) 2003-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __PARSE_CMD_H
00018 #define __PARSE_CMD_H
00019
00020 #include <stdarg.h>
00021 #include <l4/sys/compiler.h>
00022
00028
00032 enum parse_cmd_type {
00033 PARSE_CMD_INT,
00034 PARSE_CMD_SWITCH,
00035 PARSE_CMD_STRING,
00036 PARSE_CMD_FN,
00037 PARSE_CMD_FN_ARG,
00038 PARSE_CMD_INC,
00039 PARSE_CMD_DEC,
00040 };
00041
00045 typedef L4_CV void (*parse_cmd_fn_t)(int);
00046
00050 typedef L4_CV void (*parse_cmd_fn_arg_t)(int, const char*, int);
00051
00052 EXTERN_C_BEGIN
00053
00140 L4_CV int parse_cmdline(int *argc, const char***argv, char arg0, ...);
00141 L4_CV int parse_cmdlinev(int *argc, const char***argv, char arg0, va_list va);
00142 L4_CV int parse_cmdline_extra(const char*argv0, const char*line, char delim,
00143 char arg0,...);
00144
00145 EXTERN_C_END
00148 #endif
00149

```

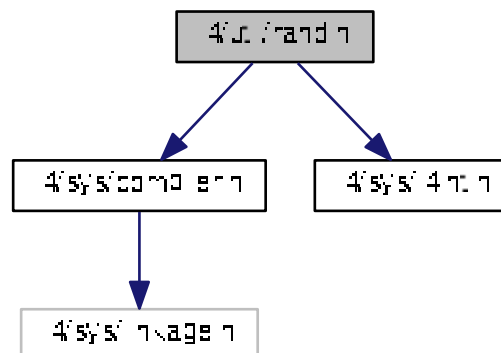
## 15.425 l4/util/rand.h File Reference

Simple Pseudo-Random Number Generator.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for rand.h:





## Functions

- [l4\\_uint32\\_t l4util\\_rand](#) (void)  
*Deliver next random number.*
- void [l4util\\_srand](#) ([l4\\_uint32\\_t](#) seed)  
*Initialize random number generator.*

### 15.425.1 Detailed Description

Simple Pseudo-Random Number Generator.

#### Date

1998

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [rand.h](#).

## 15.426 rand.h

```

00001
00008 /*
00009 * (c) 2008-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef __L4UTIL_RAND_H
00017 #define __L4UTIL_RAND_H
00018
00019 #define L4_RAND_MAX 65535
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/sys/l4int.h>
00023
00024 EXTERN_C_BEGIN
00025
00037 L4_CV l4_uint32_t
00038 l4util_rand(void);
00039
00046 L4_CV void
00047 l4util_srand (l4_uint32_t seed);
00048
00049 EXTERN_C_END
00050
00051 #endif /* __L4UTIL_RAND_H */

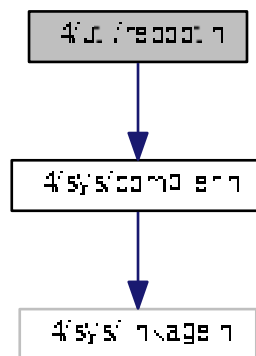
```

## 15.427 l4/util/reboot.h File Reference

Machine restarting functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for reboot.h:



### Functions

- void [l4util\\_reboot](#) (void)  
*Machine reboot.*

### 15.427.1 Detailed Description

Machine restarting functions.

Definition in file [reboot.h](#).

## 15.428 reboot.h

```

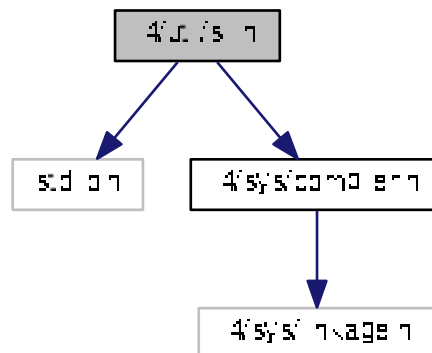
00001
00005 /*
00006 * (c) 2000-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Norman Feske <nf2@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014 /*****
00015 * #ifndef _L4UTIL_REBOOT_H
00016 * #define _L4UTIL_REBOOT_H
00017 *
00018 * #include <l4/sys/compiler.h>
00019 *
00020 * EXTERN_C_BEGIN
00021 * L4_CV void l4util_reboot(void) __attribute__((__noreturn__));
00022 * EXTERN_C_END
00023 */
00024 #endif /* !_L4UTIL_REBOOT_H */

```

## 15.429 l4/util/sll.h File Reference

List implementation.

```
#include <stdlib.h>
#include <l4/sys/compiler.h>
Include dependency graph for sll.h:
```



### 15.429.1 Detailed Description

List implementation.

Definition in file [sll.h](#).

## 15.430 sll.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Ronald Aigner <ra3@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef __L4UTIL__SLL_H__
00014 #define __L4UTIL__SLL_H__
00015
00016 #include <stdlib.h>
00017
00018 #include <l4/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00022 /*
00023 * the linked list structure
00024 */
00025
00026 typedef struct slist_t
00027 {
00028 struct slist_t *next; /* pointer to next node */
00029 void *data; /* void pointer for user data */
00030 }
```

```

00030 } slist_t;
00031
00032 /*
00033 * function prototypes
00034 */
00035 static inline slist_t*
00036 list_new_entry(void *data);
00037
00038 static inline slist_t*
00039 list_append(slist_t *list, slist_t *new_node);
00040
00041 static inline slist_t*
00042 list_remove(slist_t *list, slist_t *node);
00043
00044 static inline void
00045 list_free_entry(slist_t **list);
00046
00047 static inline unsigned char
00048 list_is_empty(slist_t *list);
00049
00050 static inline slist_t*
00051 list_get_at(slist_t *list, int n);
00052
00053 static inline slist_t*
00054 list_add(slist_t *list, slist_t *new_node);
00055
00056 static inline void
00057 list_insert_after(slist_t *after, slist_t *new_node);
00058
00059 static inline int
00060 list_elements(slist_t *head);
00061
00062 /*
00063 * allocateNode()
00064 * allocate a new node.
00065 *
00066 * Parameters:
00067 * void *data a generic pointer to object data
00068 *
00069 * Return Values:
00070 * pointer to slist_t if succeeds
00071 * NULL otherwise
00072 */
00073
00074 static inline slist_t*
00075 list_new_entry(void *data)
00076 {
00077 slist_t *sll;
00078
00079 sll = (slist_t *)malloc(sizeof(slist_t));
00080 if (!sll)
00081 return ((slist_t *) NULL);
00082
00083 sll->data=data;
00084 sll->next=NULL;
00085
00086 return (sll);
00087 }
00088
00089 /*
00090 * appendNode()
00091 * appends a node to the end of a list
00092 *
00093 * Parameters:
00094 * slist_t *head - modify the list
00095 * slist_t *new - appends this node
00096 *
00097 * Return Values:
00098 * the new list
00099 */
00100
00101 static inline slist_t*
00102 list_append(slist_t *head, slist_t *new_node)
00103 {
00104 slist_t *ret = head;
00105 if (!head)
00106 return new_node;
00107
00108 while (head->next)
00109 head = head->next;
00110 head->next = new_node;
00111 return ret;
00112 }
00113
00114 /*
00115 * insertNode()
00116 * insert a node at the beginning of a list

```

```

00117 *
00118 * Parameters:
00119 * slist_t *head - modify this list
00120 * slist_t *new - appends this node
00121 *
00122 * Return Values:
00123 * the new list
00124 *
00125 */
00126 static inline slist_t*
00127 list_add(slist_t *head, slist_t *new_node)
00128 {
00129 if (!new_node)
00130 return head;
00131 new_node->next = head;
00132 return new_node;
00133 }
00134
00135 /*
00136 * insertNode()
00137 * insert a node at the beginning of a list
00138 *
00139 * Parameters:
00140 * slist_t *head - modify this list
00141 * slist_t *new - appends this node
00142 *
00143 * Return Values:
00144 * the new list
00145 *
00146 */
00147 static inline void
00148 list_insert_after(slist_t *after, slist_t *new_node)
00149 {
00150 if (!new_node)
00151 return;
00152 if (!after)
00153 return;
00154 new_node->next = after->next;
00155 after->next = new_node;
00156 }
00157
00158
00159 /*
00160 * emptyList()
00161 * check if a list variable is NULL
00162 *
00163 * Parameters:
00164 * slist_t *list list
00165 *
00166 * Return Values:
00167 * TRUE if empty
00168 * FALSE if not empty
00169 *
00170 */
00171 static inline unsigned char
00172 list_is_empty(slist_t *list)
00173 {
00174 return ((list) ? 0 : 1);
00175 }
00176
00177 /*
00178 * delNode()
00179 * remove a node from a list
00180 *
00181 * Parameters:
00182 * slist_t *head - list to modify
00183 * slist_t *node - node to remove
00184 *
00185 * Return Values:
00186 * none
00187 *
00188 */
00189 static inline slist_t*
00190 list_remove(slist_t *head, slist_t *node)
00191 {
00192 slist_t *ret = head;
00193 if (list_is_empty(head))
00194 return ret;
00195 if (!node)
00196 return ret;
00197
00198 if (head == node)
00199 {
00200 ret = head->next;
00201 }
00202 else
00203 {

```

```

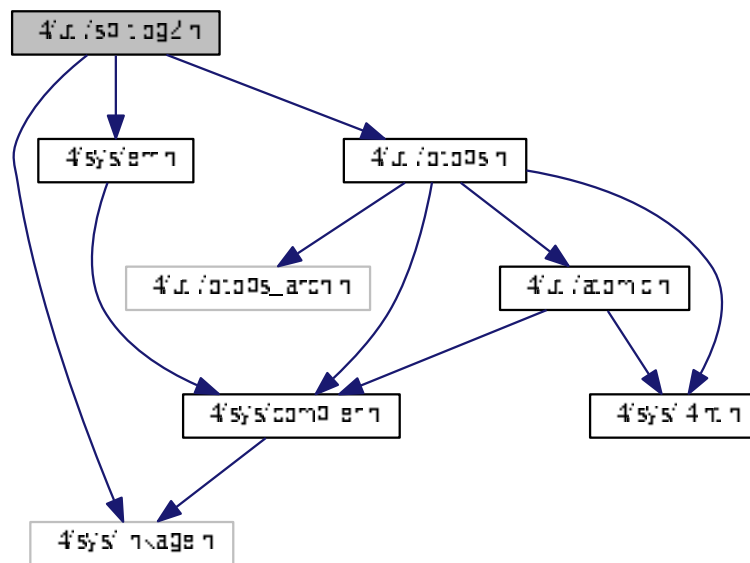
00204 while (head && (head->next != node))
00205 head = head->next;
00206 if (!head)
00207 return ret;
00208 else
00209 head->next = node->next;
00210 }
00211 list_free_entry(&node);
00212 return ret;
00213 }
00214
00215 /*
00216 * freeNode()
00217 * frees a node
00218 *
00219 * Parameters:
00220 * slist_t *list node to free
00221 *
00222 * Return Values:
00223 * none
00224 *
00225 */
00226 static inline void
00227 list_free_entry(slist_t **list)
00228 {
00229 if (*list)
00230 {
00231 free((void *) (*list));
00232 (*list)=NULL;
00233 }
00234 }
00235
00236
00237 /*
00238 * getNthNode()
00239 * get nth node in a list
00240 *
00241 * Parameters:
00242 * slist_t *list - the head list
00243 * int n - return the node
00244 * Return Values:
00245 * a pointer to the list at position n
00246 * NULL if there's no such node at posion n
00247 *
00248 */
00249 static inline slist_t*
00250 list_get_at(slist_t *list, int n)
00251 {
00252 int j=0;
00253
00254 while (list)
00255 {
00256 j++;
00257 if (j == n)
00258 return (list);
00259 list = list->next;
00260 }
00261
00262 return ((slist_t *) NULL);
00263 }
00264
00265 /*
00266 * numNodes()
00267 * returns number of nodes in the list
00268 *
00269 * Parameters:
00270 * slist_t *head - the head node of the list
00271 *
00272 * Return Values:
00273 * number of node/s
00274 *
00275 */
00276 static inline int
00277 list_elements(slist_t *head)
00278 {
00279 register int n;
00280 for (n=0; head; head=head->next) n++;
00281 return (n);
00282 }
00283
00284 EXTERN_C_END
00285
00286 #endif /* __L4UTIL__SLL_H__ */

```

## 15.431 l4/util/splitlog2.h File Reference

Split a range in log2 aligned and size-aligned chunks.

```
#include <l4/sys/linkage.h>
#include <l4/sys/err.h>
#include <l4/util/bitops.h>
Include dependency graph for splitlog2.h:
```



### Functions

- `long l4util_splitlog2_hdl (l4_addr_t start, l4_addr_t end, long(*handler)(l4_addr_t s, l4_addr_t e, int log2size))`  
Split a range into log2 base and size aligned chunks.
- `l4_addr_t l4util_splitlog2_size (l4_addr_t start, l4_addr_t end)`  
Return log2 base and size aligned length of a range.

### 15.431.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file [splitlog2.h](#).

## 15.432 splitlog2.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__INCLUDE__SPLITLOG2_H__
00013 #define __L4UTIL__INCLUDE__SPLITLOG2_H__
00014
00015 #include <l4/sys/linkage.h>
00016 #include <l4/sys/err.h>
00017 #include <l4/util/bitops.h>
00018
00019 EXTERN_C_BEGIN
00020
00021 L4_INLINE long
00022 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00023 long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));
00024
00025 L4_INLINE l4_addr_t
00026 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);
00027
00028 EXTERN_C_END
00029
00030 /* Implementation */
00031
00032 L4_INLINE long
00033 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00034 long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
00035 {
00036 if (end < start)
00037 return -L4_EINVAL;
00038 while (start <= end)
00039 {
00040 long retval;
00041 int len2 = l4util_splitlog2_size(start, end);
00042 l4_addr_t len = 1UL << len2;
00043 if ((retval = handler(start, start + len - 1, len2)))
00044 return retval;
00045 start += len;
00046 }
00047 return 0;
00048 }
00049
00050 L4_INLINE l4_addr_t
00051 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00052 {
00053 int start_bits = l4util_bsf(start);
00054 int len_bits = l4util_bsr(end - start + 1);
00055 if (start_bits != -1 && len_bits > start_bits)
00056 len_bits = start_bits;
00057 return len_bits;
00058 }
00059
00060 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */

```

## 15.433 l4/util/stack.h File Reference

Some helper functions for stack manipulation.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/util/stack_impl.h>

```



[illegible]

- `l4_addr_t l4util_stack_get_sp` (void)  
*Get current stack pointer.*

Some helper functions for stack manipulation.

```
*-((l4_threadid_t)esp) = tid
```

03/2004

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

## 15.433.2 Function Documentation

### 15.433.2.1 l4util\_stack\_get\_sp()

```
l4_addr_t l4util_stack_get_sp (
 void) [inline]
```

Get current stack pointer.

#### Returns

stack pointer.

Definition at line 23 of file [stack\\_impl.h](#).

References [EXTERN\\_C\\_END](#).

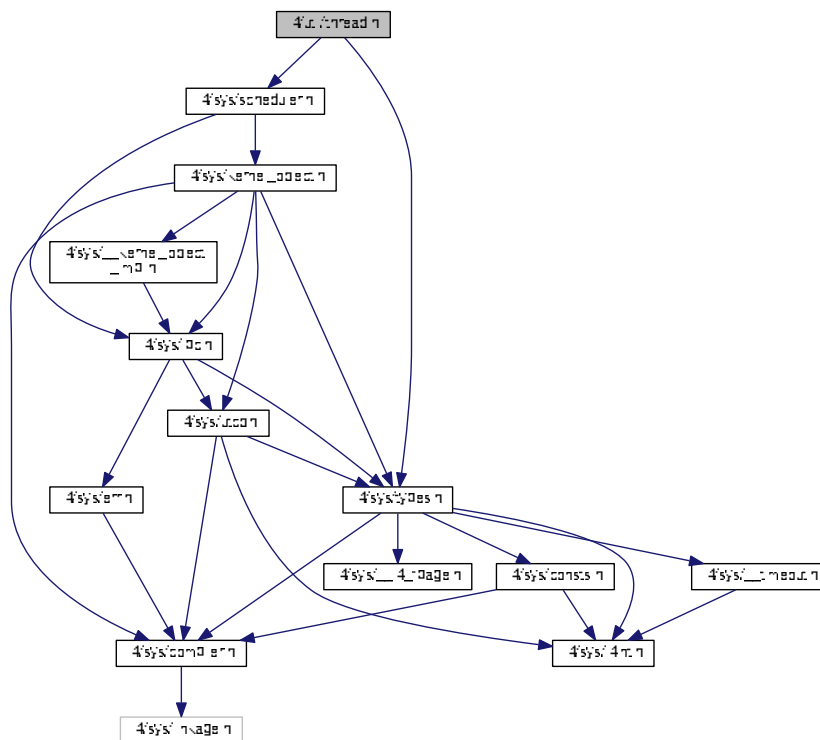
## 15.434 stack.h

```
00001
00012 /*
00013 * (c) 2003-2009 Author(s)
00014 * economic rights: Technische Universität Dresden (Germany)
00015 * This file is part of TUD:OS and distributed under the terms of the
00016 * GNU Lesser General Public License 2.1.
00017 * Please see the COPYING-LGPL-2.1 file for details.
00018 */
00019
00020 #ifndef _L4UTIL_STACK_H
00021 #define _L4UTIL_STACK_H
00022
00023 #include <l4/sys/types.h>
00024 #include <l4/sys/compiler.h>
00025
00026 EXTERN_C_BEGIN
00027
00028 L4_INLINE void l4util_stack_push_mword(l4_addr_t *stack, l4_mword_t val);
00029
00030 /*****
00036 L4_INLINE l4_addr_t l4util_stack_get_sp(void);
00037
00038 */
00039 * Implementations.
00040 */
00041
00042 #include <l4/util/stack_impl.h>
00043
00044 L4_INLINE void
00045 l4util_stack_push_mword(l4_addr_t *stack, l4_mword_t val)
00046 {
00047 l4_mword_t *esp = (l4_mword_t*)(*stack);
00048 *--esp = val;
00049 *stack = (l4_addr_t)esp;
00050 }
00051
00052 EXTERN_C_END
00053
00054 #endif
```

## 15.435 l4/util/thread.h File Reference

Low-level Thread Functions.

```
#include <l4/sys/types.h>
#include <l4/sys/scheduler.h>
Include dependency graph for thread.h:
```



## Macros

- `#define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)`  
*Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.*

### 15.435.1 Detailed Description

### Low-level Thread Functions.

Date \_\_\_\_\_

1997

**Author**

Sebastian Schönberg

Definition in file [thread.h](#).

## 15.435.2 Macro Definition Documentation

### 15.435.2.1 \_\_L4UTIL\_THREAD\_FUNC

```
#define __L4UTIL_THREAD_FUNC(
 name) void L4_NORETURN name(void)
```

Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.

Use this as a function header when starting a low-level thread where only stack and instruction pointer are in a well-defined state.

Example:

```
L4UTIL_THREAD_FUNC(helper_thread) { for(;;); }
```

```
thread_cap->ex_regs((l4_umword_t)helper_thread, stack_addr);
```

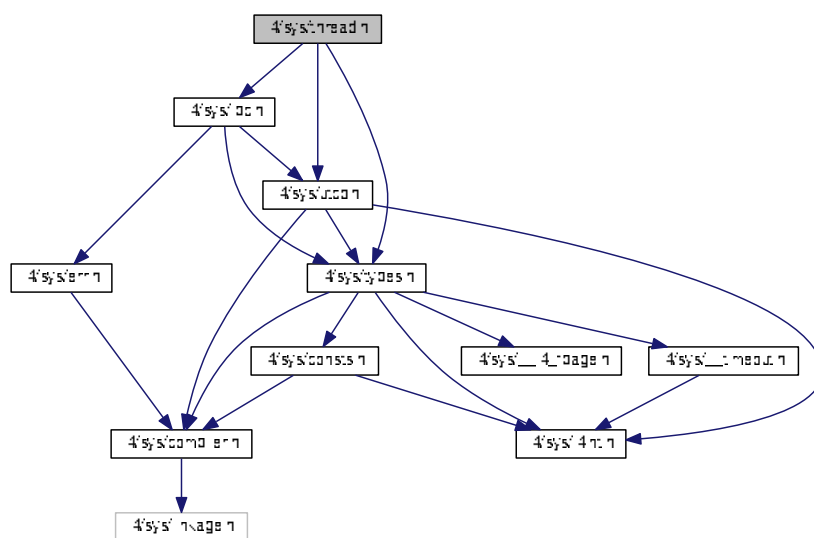
Definition at line 72 of file [thread.h](#).

## 15.436 thread.h

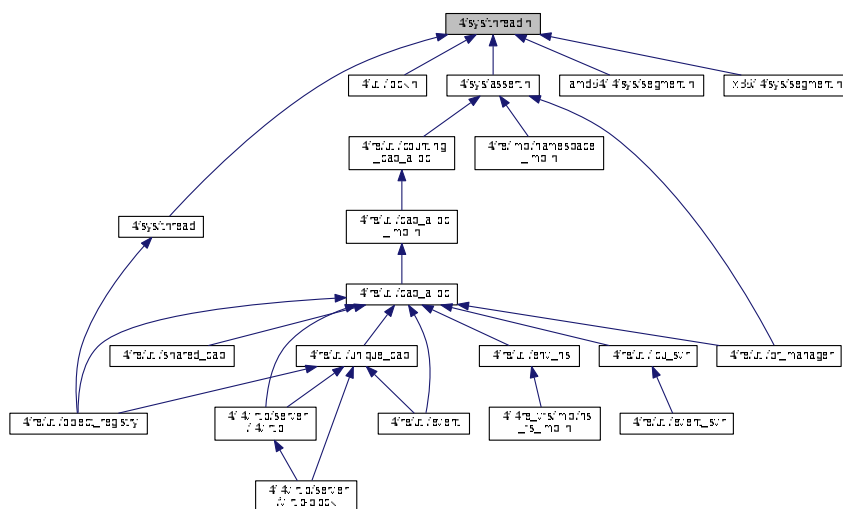
```
00001
00008 /*
00009 * (c) 2003-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef __L4_THREAD_H
00017 #define __L4_THREAD_H
00018
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/scheduler.h>
00021
00022 EXTERN_C_BEGIN
00023
00046 L4_CV long
00047 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00048 l4_cap_idx_t factory,
00049 l4_umword_t pc, l4_umword_t sp,
00050 l4_cap_idx_t pager,
00051 l4_cap_idx_t task,
00052 l4_cap_idx_t scheduler, l4_sched_param_t scp)
00053 L4_NOTHROW;
00054
00055 EXTERN_C_END
00056
00055 #ifndef L4UTIL_THREAD_FUNC
00056
00072 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00073 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00074 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00075 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00076 #endif
00077
00078 #endif /* __L4_THREAD_H */
```

## Common thread related definitions.

Include dependency graph for thread.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_thread_ops` {  
`L4_THREAD_CONTROL_OP` = 0UL, `L4_THREAD_EX_REGS_OP` = 1UL, `L4_THREAD_SWITCH_OP` = 2UL,  
`L4_THREAD_STATS_OP` = 3UL,  
`L4_THREAD_VCPU_RESUME_OP` = 4UL, `L4_THREAD_REGISTER_DELETE_IRQ_OP` = 5UL, `L4_THREAD_READ_MODIFY_SENDER_OP` = 6UL, `L4_THREAD_VCPU_CONTROL_OP` = 7UL,  
`L4_THREAD_X86_GDT_OP` = 0x10UL, `L4_THREAD_ARM_TPIDRURO_OP` = 0x10UL, `L4_THREAD_AMD64_SET_SEGMENT_BASE_OP` = 0x12UL, `L4_THREAD_AMD64_GET_SEGMENT_INFO_OP` = 0x13UL,  
`L4_THREAD_OPCODE_MASK` = 0xffff }  
*Operations on thread objects.*
- enum `L4_thread_control_flags` {  
`L4_THREAD_CONTROL_SET_PAGER` = 0x0010000, `L4_THREAD_CONTROL_BIND_TASK` = 0x0200000,  
`L4_THREAD_CONTROL_ALIEN` = 0x0400000, `L4_THREAD_CONTROL_UX_NATIVE` = 0x0800000,  
`L4_THREAD_CONTROL_SET_EXC_HANDLER` = 0x1000000 }  
*Flags for the thread control operation.*
- enum `L4_thread_control_mr_indices` {  
`L4_THREAD_CONTROL_MR_IDX_FLAGS` = 0, `L4_THREAD_CONTROL_MR_IDX_PAGER` = 1, `L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER` = 2, `L4_THREAD_CONTROL_MR_IDX_FLAG_VALS` = 4,  
`L4_THREAD_CONTROL_MR_IDX_BIND_UTCB` = 5, `L4_THREAD_CONTROL_MR_IDX_BIND_TASK` = 6  
}
- enum `L4_thread_ex_regs_flags` { `L4_THREAD_EX_REGS_CANCEL` = 0x10000UL, `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` = 0x20000UL }  
*Flags for the thread ex-regs operation.*

## Functions

- `l4_msgtag_t l4_thread_ex_regs (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags) L4_NOTHROW`  
*Exchange basic thread registers.*
- `l4_msgtag_t l4_thread_ex_regs_u (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW`  
*Exchange basic thread registers.*
- `l4_msgtag_t l4_thread_ex_regs_ret (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags) L4_NOTHROW`  
*Exchange basic thread registers and return previous values.*
- `l4_msgtag_t l4_thread_ex_regs_ret_u (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW`  
*Exchange basic thread registers and return previous values.*
- void `l4_thread_control_start (void) L4_NOTHROW`  
*Start a thread control API sequence.*
- void `l4_thread_control_pager (l4_cap_idx_t pager) L4_NOTHROW`  
*Set the pager.*
- void `l4_thread_control_exc_handler (l4_cap_idx_t exc_handler) L4_NOTHROW`  
*Set the exception handler.*
- void `l4_thread_control_bind (l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW`  
*Bind the thread to a task.*
- void `l4_thread_control_alien (int on) L4_NOTHROW`  
*Enable alien mode.*
- void `l4_thread_control_ux_host_syscall (int on) L4_NOTHROW`  
*Enable pass through of native host (Linux) system calls.*

- `l4_msgtag_t l4_thread_control_commit (l4_cap_idx_t thread) L4_NOTHROW`  
*Commit the thread control parameters.*
- `l4_msgtag_t l4_thread_yield (void) L4_NOTHROW`  
*Yield current time slice.*
- `l4_msgtag_t l4_thread_switch (l4_cap_idx_t to_thread) L4_NOTHROW`  
*Switch to another thread (and donate the remaining time slice).*
- `l4_msgtag_t l4_thread_stats_time (l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW`  
*Get consumed time of thread in  $\mu$ s.*
- `l4_msgtag_t l4_thread_vcpu_resume_start (void) L4_NOTHROW`  
*vCPU return from event handler.*
- `l4_msgtag_t l4_thread_vcpu_resume_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW`  
*Commit vCPU resume.*
- `l4_msgtag_t l4_thread_vcpu_control (l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW`  
*Enable or disable the vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_vcpu_control_u (l4_cap_idx_t thread, l4_addr_t vcpu_state, l4_utcb_t *utcb) L4_NOTHROW`  
*Enable or disable the vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_vcpu_control_ext (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW`  
*Enable or disable the extended vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_vcpu_control_ext_u (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state, l4_utcb_t *utcb) L4_NOTHROW`  
*Enable or disable the extended vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_register_del_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW`  
*Register an IRQ that will trigger upon deletion events.*
- `l4_msgtag_t l4_thread_modify_sender_start (void) L4_NOTHROW`  
*Start a thread sender modification sequence.*
- `int l4_thread_modify_sender_add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_msgtag_t *tag) L4_NOTHROW`  
*Add a modification pattern to a sender modification sequence.*
- `l4_msgtag_t l4_thread_modify_sender_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW`  
*Apply (commit) a sender modification sequence.*

### 15.437.1 Detailed Description

Common thread related definitions.

Definition in file [thread.h](#).

## 15.438 thread.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate

```

```

00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/utcb.h>
00029 #include <l4/sys/ipc.h>
00030
00087 L4_INLINE l4_msgtag_t
00088 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip,
00089 l4_addr_t sp,
00090 l4_umword_t flags) L4_NOTHROW;
00091
00097 L4_INLINE l4_msgtag_t
00098 l4_thread_ex_regs_u(l4_cap_idx_t thread,
00099 l4_addr_t ip, l4_addr_t sp,
00100 l4_umword_t flags, l4_utcb_t *utcb)
00101 L4_NOTHROW;
00102
00124 L4_INLINE l4_msgtag_t
00125 l4_thread_ex_regs_ret(l4_cap_idx_t thread,
00126 l4_addr_t *ip, l4_addr_t *sp,
00127 l4_umword_t *flags) L4_NOTHROW;
00128
00134 L4_INLINE l4_msgtag_t
00135 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread,
00136 l4_addr_t *ip, l4_addr_t *sp,
00137 l4_umword_t *flags, l4_utcb_t *utcb)
00138 L4_NOTHROW;
00139
00187 L4_INLINE void
00188 l4_thread_control_start(void) L4_NOTHROW;
00189
00194 L4_INLINE void
00195 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00196
00206 L4_INLINE void
00207 l4_thread_control_pager(l4_cap_idx_t pager)
00208 L4_NOTHROW;
00209
00213 L4_INLINE void
00214 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb)
00215 L4_NOTHROW;
00216
00225 L4_INLINE void
00226 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler)
00227 L4_NOTHROW;
00228
00232 L4_INLINE void
00233 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00234 l4_utcb_t *utcb) L4_NOTHROW;
00235
00251 L4_INLINE void
00252 l4_thread_control_bind(l4_utcb_t *thread_utcb,
00253 l4_cap_idx_t task) L4_NOTHROW;
00254
00259 L4_INLINE void
00260 l4_thread_control_bind_u(l4_utcb_t *thread_utcb,
00261 l4_cap_idx_t task, l4_utcb_t *utcb)
00262 L4_NOTHROW;
00263
00277 L4_INLINE void
00278 l4_thread_control_alien(int on) L4_NOTHROW;
00279
00284 L4_INLINE void
00285 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00286
00297 L4_INLINE void
00298 l4_thread_control_ux_host_syscall(int on)
00299 L4_NOTHROW;
00300
00304 L4_INLINE void
00305 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00306
00307
00308
00316 L4_INLINE l4_msgtag_t
00317 l4_thread_control_commit(l4_cap_idx_t thread)
00318 L4_NOTHROW;

```



```

00323 L4_INLINE l4_msgtag_t
00324 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb)
 L4_NOTHROW;
00325
00332 L4_INLINE l4_msgtag_t
00333 l4_thread_yield(void) L4_NOTHROW;
00334
00343 L4_INLINE l4_msgtag_t
00344 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW;
00345
00350 L4_INLINE l4_msgtag_t
00351 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb)
 L4_NOTHROW;
00352
00353
00354
00364 L4_INLINE l4_msgtag_t
00365 l4_thread_stats_time(l4_cap_idx_t thread,
 l4_kernel_clock_t *us) L4_NOTHROW;
00366
00371 L4_INLINE l4_msgtag_t
00372 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00373 l4_utcb_t *utcb) L4_NOTHROW;
00374
00375
00386 L4_INLINE l4_msgtag_t
00387 l4_thread_vcpu_resume_start(void) L4_NOTHROW;
00388
00393 L4_INLINE l4_msgtag_t
00394 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00395
00422 L4_INLINE l4_msgtag_t
00423 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00424 l4_msgtag_t tag) L4_NOTHROW;
00425
00430 L4_INLINE l4_msgtag_t
00431 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00432 l4_msgtag_t tag, l4_utcb_t *utcb)
 L4_NOTHROW;
00433
00434
00451 L4_INLINE l4_msgtag_t
00452 l4_thread_vcpu_control(l4_cap_idx_t thread,
 l4_addr_t vcpu_state) L4_NOTHROW;
00453
00461 L4_INLINE l4_msgtag_t
00462 l4_thread_vcpu_control_u(l4_cap_idx_t thread,
 l4_addr_t vcpu_state,
00463 l4_utcb_t *utcb) L4_NOTHROW;
00464
00487 L4_INLINE l4_msgtag_t
00488 l4_thread_vcpu_control_ext(l4_cap_idx_t thread,
 l4_addr_t ext_vcpu_state) L4_NOTHROW;
00489
00497 L4_INLINE l4_msgtag_t
00498 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread,
 l4_addr_t ext_vcpu_state,
00499 l4_utcb_t *utcb) L4_NOTHROW;
00500
00501
00514 L4_INLINE l4_msgtag_t
00515 l4_thread_register_del_irq(l4_cap_idx_t thread,
 l4_cap_idx_t irq) L4_NOTHROW;
00516
00521 L4_INLINE l4_msgtag_t
00522 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00523 l4_utcb_t *utcb) L4_NOTHROW;
00524
00536 L4_INLINE l4_msgtag_t
00537 l4_thread_modify_sender_start(void) L4_NOTHROW;
00538
00543 L4_INLINE l4_msgtag_t
00544 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW;
00545
00570 L4_INLINE int
00571 l4_thread_modify_sender_add(l4_umword_t match_mask,
00572 l4_umword_t match,
00573 l4_umword_t del_bits,
00574 l4_umword_t add_bits,
00575 l4_msgtag_t *tag) L4_NOTHROW;
00576
00581 L4_INLINE int
00582 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
00583 l4_umword_t match,
00584 l4_umword_t del_bits,
00585 l4_umword_t add_bits,
00586 l4_msgtag_t *tag, l4_utcb_t *u)

```

```

 L4_NOTHROW;
00587
00595 L4_INLINE l4_msgtag_t
00596 l4_thread_modify_sender_commit(l4_cap_idx_t thread,
 l4_msgtag_t tag) L4_NOTHROW;
00597
00602 L4_INLINE l4_msgtag_t
00603 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
00604 l4_utcb_t *u) L4_NOTHROW;
00605
00612 enum L4_thread_ops
00613 {
00614 L4_THREAD_CONTROL_OP = 0UL,
00615 L4_THREAD_EX_REGS_OP = 1UL,
00616 L4_THREAD_SWITCH_OP = 2UL,
00617 L4_THREAD_STATS_OP = 3UL,
00618 L4_THREAD_VCPU_RESUME_OP = 4UL,
00619 L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL,
00620 L4_THREAD_MODIFY_SENDER_OP = 6UL,
00621 L4_THREAD_VCPU_CONTROL_OP = 7UL,
00622 L4_THREAD_VCPU_CONTROL_EXT_OP = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
00623 L4_THREAD_X86_GDT_OP = 0x10UL,
00624 L4_THREAD_ARM_TPIDRURO_OP = 0x10UL,
00625 L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
00626 L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
00627 L4_THREAD_OPCODE_MASK = 0xffff,
00628 };
00629
00640 enum L4_thread_control_flags
00641 {
00643 L4_THREAD_CONTROL_SET_PAGER = 0x00100000,
00645 L4_THREAD_CONTROL_BIND_TASK = 0x02000000,
00647 L4_THREAD_CONTROL_ALIEN = 0x04000000,
00649 L4_THREAD_CONTROL_UX_NATIVE = 0x08000000,
00651 L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x10000000,
00652 };
00653
00663 enum L4_thread_control_mr_indices
00664 {
00665 L4_THREAD_CONTROL_MR_IDX_FLAGS = 0,
00666 L4_THREAD_CONTROL_MR_IDX_PAGER = 1,
00667 L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2,
00668 L4_THREAD_CONTROL_MR_IDX_FLAG_VALS = 4,
00669 L4_THREAD_CONTROL_MR_IDX_BIND_UTCB = 5,
00670 L4_THREAD_CONTROL_MR_IDX_BIND_TASK = 6,
00671 };
00672
00678 enum L4_thread_ex_regs_flags
00679 {
00680 L4_THREAD_EX_REGS_CANCEL = 0x10000UL,
00681 L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL,
00682 };
00683
00684
00685 /* IMPLEMENTATION -----*/
00686
00687 #include <l4/sys/ipc.h>
00688 #include <l4/sys/types.h>
00689
00690 L4_INLINE l4_msgtag_t
00691 l4_thread_ex_regs_u(l4_cap_idx_t thread,
 l4_addr_t ip, l4_addr_t sp,
00692 l4_umword_t flags, l4_utcb_t *utcb)
 L4_NOTHROW
00693 {
00694 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00695 v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
00696 v->mr[1] = ip;
00697 v->mr[2] = sp;
00698 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0),
 L4_IPC_NEVER);
00699 }
00700
00701 L4_INLINE l4_msgtag_t
00702 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread,
 l4_addr_t *ip, l4_addr_t *sp,
00703 l4_umword_t *flags, l4_utcb_t *utcb)
 L4_NOTHROW
00704 {
00705 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00706 l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
00707 if (l4_error_u(ret, utcb))
00708 return ret;
00709
00710 *flags = v->mr[0];
00711 *ip = v->mr[1];
00712 *sp = v->mr[2];

```

```

00713 return ret;
00714 }
00715
00716 L4_INLINE void
00717 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
00718 {
00719 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00720 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] =
00721 L4_THREAD_CONTROL_OP;
00722 }
00723 L4_INLINE void
00724 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb)
00725 L4_NOTHROW
00726 {
00727 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00728 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00729 L4_THREAD_CONTROL_SET_PAGER;
00730 v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER] = pager;
00731 }
00732 L4_INLINE void
00733 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00734 l4_utcb_t *utcb) L4_NOTHROW
00735 {
00736 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00737 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00738 L4_THREAD_CONTROL_SET_EXC_HANDLER;
00739 v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER] = exc_handler;
00740 }
00741 L4_INLINE void
00742 l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
00743 l4_utcb_t *utcb) L4_NOTHROW
00744 {
00745 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00746 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00747 L4_THREAD_CONTROL_BIND_TASK;
00748 v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB] = (
00749 l4_addr_t)thread_utcb;
00750 v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK] =
00751 L4_ITEM_MAP;
00752 v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1] =
00753 l4_obj_fpage(task, 0, L4_FPAGE_RWX).raw;
00754 }
00755 L4_INLINE void
00756 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00757 {
00758 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00759 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00760 L4_THREAD_CONTROL_ALIEN;
00761 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ?
00762 L4_THREAD_CONTROL_ALIEN : 0;
00763 }
00764 L4_INLINE void
00765 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00766 {
00767 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00768 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00769 L4_THREAD_CONTROL_UX_NATIVE;
00770 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ?
00771 L4_THREAD_CONTROL_UX_NATIVE : 0;
00772 }
00773 L4_INLINE l4_msgtag_t
00774 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb)
00775 L4_NOTHROW
00776 {
00777 int items = 0;
00778 if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] &
00779 L4_THREAD_CONTROL_BIND_TASK)
00780 items = 1;
00781 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items,
00782 0), L4_IPC_NEVER);
00783 }
00784 L4_INLINE l4_msgtag_t
00785 l4_thread_yield(void) L4_NOTHROW
00786 {
00787 l4_ipc_receive(L4_INVALID_CAP, NULL,
00788 L4_IPC_BOTH_TIMEOUT_0);
00789 return l4_msgtag(0, 0, 0, 0);
00790 }
00791
00792
00793

```

```

00784 /* Preliminary, to be changed */
00785 L4_INLINE l4_msgtag_t
00786 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb)
00787 L4_NOTHROW
00788 {
00789 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00790 v->mr[0] = L4_THREAD_SWITCH_OP;
00791 return l4_ipc_call(to_thread, utcb, l4_msgtag(
00792 L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00793 }
00794 L4_INLINE l4_msgtag_t
00795 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00796 l4_utcb_t *utcb) L4_NOTHROW
00797 {
00798 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00799 l4_msgtag_t res;
00800
00801 v->mr[0] = L4_THREAD_STATS_OP;
00802
00803 res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
00804 L4_IPC_NEVER);
00805
00806 if (l4_msgtag_has_error(res))
00807 return res;
00808
00809 *us = v->mr64[l4_utcb_mr64_idx(0)];
00810 return res;
00811 }
00812 L4_INLINE l4_msgtag_t
00813 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
00814 {
00815 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00816 v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
00817 return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00818 }
00819 L4_INLINE l4_msgtag_t
00820 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00821 l4_msgtag_t tag, l4_utcb_t *utcb)
00822 L4_NOTHROW
00823 {
00824 return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
00825 }
00826 L4_INLINE l4_msgtag_t
00827 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip,
00828 l4_addr_t sp,
00829 l4_umword_t flags) L4_NOTHROW
00830 {
00831 return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
00832 }
00833 L4_INLINE l4_msgtag_t
00834 l4_thread_ex_regs_ret(l4_cap_idx_t thread,
00835 l4_addr_t *ip, l4_addr_t *sp,
00836 l4_umword_t *flags) L4_NOTHROW
00837 {
00838 return l4_thread_ex_regs_ret_u(thread, ip, sp, flags,
00839 l4_utcb());
00840 }
00841 L4_INLINE void
00842 l4_thread_control_start(void) L4_NOTHROW
00843 {
00844 l4_thread_control_start_u(l4_utcb());
00845 }
00846 L4_INLINE void
00847 l4_thread_control_pager(l4_cap_idx_t pager)
00848 L4_NOTHROW
00849 {
00850 l4_thread_control_pager_u(pager, l4_utcb());
00851 }
00852 L4_INLINE void
00853 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler)
00854 L4_NOTHROW
00855 {
00856 l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
00857 }
00858 L4_INLINE void
00859
00860
00861 L4_INLINE void

```

```

00862 l4_thread_control_bind(l4_utcb_t *thread_utcb,
 l4_cap_idx_t task) L4_NOTHROW
00863 {
00864 l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
00865 }
00866
00867 L4_INLINE void
00868 l4_thread_control_alien(int on) L4_NOTHROW
00869 {
00870 l4_thread_control_alien_u(l4_utcb(), on);
00871 }
00872
00873 L4_INLINE void
00874 l4_thread_control_ux_host_syscall(int on)
 L4_NOTHROW
00875 {
00876 l4_thread_control_ux_host_syscall_u(l4_utcb(), on);
00877 }
00878
00879 L4_INLINE l4_msgtag_t
00880 l4_thread_control_commit(l4_cap_idx_t thread)
 L4_NOTHROW
00881 {
00882 return l4_thread_control_commit_u(thread, l4_utcb());
00883 }
00884
00885
00886
00887
00888 L4_INLINE l4_msgtag_t
00889 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
00890 {
00891 return l4_thread_switch_u(to_thread, l4_utcb());
00892 }
00893
00894
00895
00896
00897 L4_INLINE l4_msgtag_t
00898 l4_thread_stats_time(l4_cap_idx_t thread,
 l4_kernel_clock_t *us) L4_NOTHROW
00899 {
00900 return l4_thread_stats_time_u(thread, us, l4_utcb());
00901 }
00902
00903 L4_INLINE l4_msgtag_t
00904 l4_thread_vcpu_resume_start(void) L4_NOTHROW
00905 {
00906 return l4_thread_vcpu_resume_start_u(l4_utcb());
00907 }
00908
00909 L4_INLINE l4_msgtag_t
00910 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
 l4_msgtag_t tag) L4_NOTHROW
00911 {
00912 return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
00913 }
00914
00915
00916
00917 L4_INLINE l4_msgtag_t
00918 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
 l4_utcb_t *u) L4_NOTHROW
00919 {
00920 {
00921 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00922 m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
00923 m->mr[1] = l4_map_obj_control(0,0);
00924 m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).
raw;
00925 return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0),
 L4_IPC_NEVER);
00926 }
00927 }
00928
00929 L4_INLINE l4_msgtag_t
00930 l4_thread_register_del_irq(l4_cap_idx_t thread,
 l4_cap_idx_t irq) L4_NOTHROW
00931 {
00932 return l4_thread_register_del_irq_u(thread, irq, l4_utcb());
00933 }
00934
00935
00936 L4_INLINE l4_msgtag_t
00937 l4_thread_vcpu_control_u(l4_cap_idx_t thread,
 l4_addr_t vcpu_state,
 l4_utcb_t *utcb) L4_NOTHROW
00938 {
00939 {
00940 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);

```

```

00941 v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
00942 v->mr[1] = vcpu_state;
00943 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
L4_IPC_NEVER);
00944 }
00945
00946 L4_INLINE l4_msgtag_t
00947 l4_thread_vcpu_control(l4_cap_idx_t thread,
l4_addr_t vcpu_state) L4_NOTHROW
00948 { return l4_thread_vcpu_control_u(thread, vcpu_state,
l4_utcb()); }
00949
00950
00951 L4_INLINE l4_msgtag_t
00952 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread,
l4_addr_t ext_vcpu_state,
l4_utcb_t *utcb) L4_NOTHROW
00953 {
00954 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00955 v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
00956 v->mr[1] = ext_vcpu_state;
00957 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
L4_IPC_NEVER);
00958 }
00959
00960
00961 L4_INLINE l4_msgtag_t
00962 l4_thread_vcpu_control_ext(l4_cap_idx_t thread,
l4_addr_t ext_vcpu_state) L4_NOTHROW
00963 { return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state,
l4_utcb()); }
00964
00965 L4_INLINE l4_msgtag_t
00966 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
00967 {
00968 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00969 m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00970 return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00971 }
00972
00973 L4_INLINE int
00974 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
l4_umword_t match,
l4_umword_t del_bits,
l4_umword_t add_bits,
l4_msgtag_t *tag, l4_utcb_t *u)
L4_NOTHROW
00979 {
00980 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00981 unsigned w = l4_msgtag_words(*tag);
00982 if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00983 return -L4_ENOMEM;
00984
00985 m->mr[w] = match_mask;
00986 m->mr[w+1] = match;
00987 m->mr[w+2] = del_bits;
00988 m->mr[w+3] = add_bits;
00989
00990 *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);
00991
00992 return 0;
00993 }
00994
00995 L4_INLINE l4_msgtag_t
00996 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
l4_utcb_t *u) L4_NOTHROW
00997 {
00998 return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
01000 }
01001
01002 L4_INLINE l4_msgtag_t
01003 l4_thread_modify_sender_start(void) L4_NOTHROW
01004 {
01005 return l4_thread_modify_sender_start_u(l4_utcb());
01006 }
01007
01008 L4_INLINE int
01009 l4_thread_modify_sender_add(l4_umword_t match_mask,
l4_umword_t match,
l4_umword_t del_bits,
l4_umword_t add_bits,
l4_msgtag_t *tag) L4_NOTHROW
01014 {
01015 return l4_thread_modify_sender_add_u(match_mask, match,
del_bits, add_bits, tag, l4_utcb());
01016 }
01017
01018
01019 L4_INLINE l4_msgtag_t

```

```

01020 l4_thread_modify_sender_commit(l4_cap_idx_t thread,
 l4_msgtag_t tag) L4_NOTHROW
01021 {
01022 return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
01023 }

```

## 15.439 arm/l4/sys/thread.h File Reference

ARM-specific thread related definitions.

### Functions

- [l4\\_msgtag\\_t l4\\_thread\\_arm\\_set\\_tpidruro](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_addr\\_t](#) tpidruro) [L4\\_NOTHROW](#)  
Set the *TPIDRURO* thread specific register.

### 15.439.1 Detailed Description

ARM-specific thread related definitions.

Definition in file [thread.h](#).

## 15.440 thread.h

```

00001
00005 /*
00006 * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include_next <l4/sys/thread.h>
00025
00034 L4_INLINE l4_msgtag_t
00035 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread,
 l4_addr_t tpidruro) L4_NOTHROW;
00036
00041 L4_INLINE l4_msgtag_t
00042 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
 l4_utcb_t *utcb) L4_NOTHROW;
00043
00044
00045 /* IMPLEMENTATION ----- */
00046
00047 L4_INLINE l4_msgtag_t
00048 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
 l4_utcb_t *utcb) L4_NOTHROW
00049 {
00050 {
00051 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00052 v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00053 v->mr[1] = tpidruro;
00054 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
 L4_IPC_NEVER);
00055 }
00056 }
00057
00058 L4_INLINE l4_msgtag_t
00059 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread,
 l4_addr_t tpidruro) L4_NOTHROW
00060 {
00061 return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00062 }

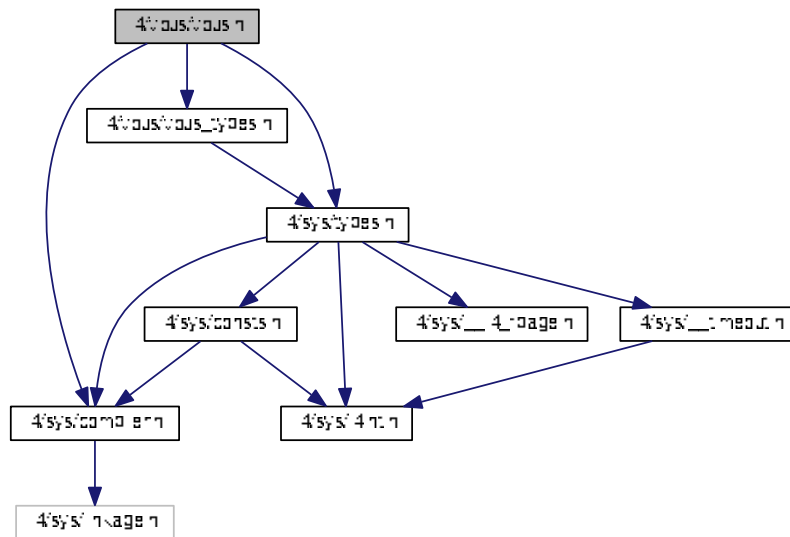
```

## 15.441 l4/vbus/vbus.h File Reference

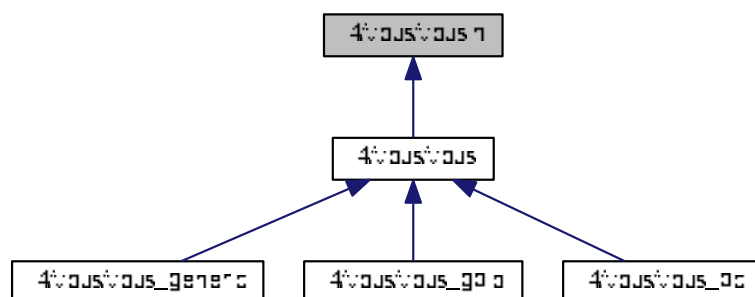
Description of the vbus C API.

```
#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>
```

Include dependency graph for vbus.h:



This graph shows which files directly or indirectly include this file:



### Enumerations

- enum { [L4VBUS\\_NULL](#) = 0, [L4VBUS\\_ROOT\\_BUS](#) = 0 }  
Constants for device nodes.
- enum [L4vbus\\_dma\\_domain\\_assign\\_flags](#) { [L4VBUS\\_DMAD\\_UNBIND](#) = 0, [L4VBUS\\_DMAD\\_BIND](#) = 1, [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#) = 0, [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#) = 2 }  
Flags for [l4vbus\\_assign\\_dma\\_domain\(\)](#).



## Functions

- int [l4vbus\\_get\\_device\\_by\\_hid](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) parent, [l4vbus\\_device\\_handle\\_t](#) \*child, char const \*hid, int depth, [l4vbus\\_device\\_t](#) \*devinfo)  
Find a device by the human interface identifier (HID).
- int [l4vbus\\_get\\_next\\_device](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) parent, [l4vbus\\_device\\_handle\\_t](#) \*child, int depth, [l4vbus\\_device\\_t](#) \*devinfo)  
Find next child following *child*.
- int [l4vbus\\_get\\_device](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, [l4vbus\\_device\\_t](#) \*devinfo)  
Obtain detailed information about a Vbus device.
- int [l4vbus\\_get\\_resource](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, int res\_idx, [l4vbus\\_resource\\_t](#) \*res)  
Obtain the resource description of an individual device resource.
- int [l4vbus\\_is\\_compatible](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, char const \*cid)  
Check if the given device has a compatibility ID (CID) or HID that matches *cid*.
- int [l4vbus\\_get\\_hid](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, char \*hid, unsigned long max\_len)  
Get the HID (hardware identifier) of a device.
- int [l4vbus\\_request\\_resource](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_resource\\_t](#) const \*res, int flags)  
Request a resource of a specific type.
- int [l4vbus\\_assign\\_dma\\_domain](#) ([l4\\_cap\\_idx\\_t](#) vbus, unsigned domain\_id, unsigned flags, [l4\\_cap\\_idx\\_t](#) dma\_space)  
Bind or unbind a kernel DMA space ([L4::Task](#)) or a [L4Re::Dma\\_space](#) to a DMA domain.
- int [l4vbus\\_release\\_resource](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_resource\\_t](#) const \*res)  
Release a previously requested resource.
- int [l4vbus\\_vicu\\_get\\_cap](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) icu, [l4\\_cap\\_idx\\_t](#) cap)  
Get capability of ICU.

### 15.441.1 Detailed Description

Description of the vbus C API.

Definition in file [vbus.h](#).

### 15.441.2 Enumeration Type Documentation

#### 15.441.2.1 anonymous enum

anonymous enum

Constants for device nodes.

Enumerator

|                 |                          |
|-----------------|--------------------------|
| L4VBUS_NULL     | NULL device.             |
| L4VBUS_ROOT_BUS | Root device on the vbus. |

Definition at line 22 of file [vbus.h](#).

## 15.442 vbus.h

```

00001 /*
00002 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00004 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 */
00015 #pragma once
00016
00017 #include <l4/sys/compiler.h>
00018 #include <l4/vbus/vbus_types.h>
00019 #include <l4/sys/types.h>
00020
00022 enum {
00023 L4VBUS_NULL = 0,
00024 L4VBUS_ROOT_BUS = 0,
00025 };
00026
00045 __BEGIN_DECLS
00046
00053 int L4_CV
00054 l4vbus_get_device_by_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t
 parent,
00055 l4vbus_device_handle_t *child, char const *hid,
00056 int depth, l4vbus_device_t *devinfo);
00057
00069 int L4_CV
00070 l4vbus_get_next_device(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00071 l4vbus_device_handle_t *child, int depth,
00072 l4vbus_device_t *devinfo);
00073
00088 int L4_CV
00089 l4vbus_get_device(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00090 l4vbus_device_t *devinfo);
00091
00100 int L4_CV
00101 l4vbus_get_resource(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00102 int res_idx, l4vbus_resource_t *res);
00103
00104
00111 int L4_CV
00112 l4vbus_is_compatible(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00113 char const *cid);
00114
00125 int L4_CV
00126 l4vbus_get_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid,
00127 unsigned long max_len);
00128
00146 int L4_CV
00147 l4vbus_request_resource(l4_cap_idx_t vbus,
00148 l4vbus_resource_t const *res,
00149 int flags);
00149
00153 enum L4vbus_dma_domain_assign_flags
00154 {
00156 L4VBUS_DMAD_UNBIND = 0,
00158 L4VBUS_DMAD_BIND = 1,
00160 L4VBUS_DMAD_L4RE_DMA_SPACE = 0,
00162 L4VBUS_DMAD_KERNEL_DMA_SPACE = 2,
00163 };
00164
00185 int L4_CV
00186 l4vbus_assign_dma_domain(l4_cap_idx_t vbus, unsigned domain_id,
00187 unsigned flags, l4_cap_idx_t dma_space);
00188
00197 int L4_CV
00198 l4vbus_release_resource(l4_cap_idx_t vbus,
00199 l4vbus_resource_t const *res);
00199
00209 int L4_CV
00210 l4vbus_vicu_get_cap(l4_cap_idx_t vbus, l4vbus_device_handle_t icu,
00211 l4_cap_idx_t cap);
00212
00213 __END_DECLS
00214

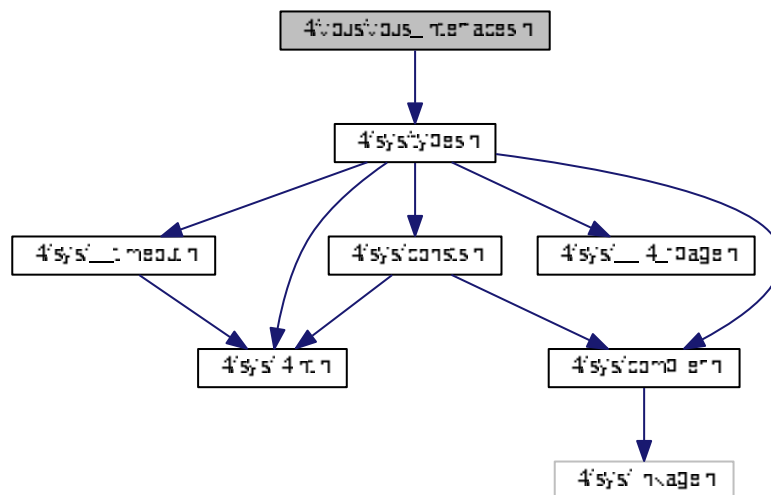
```

## 15.443 l4/vbus/vbus\_interfaces.h File Reference

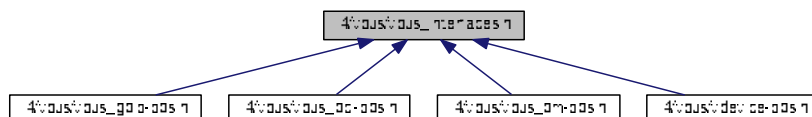
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

```
#include <l4/sys/types.h>
```

Include dependency graph for vbus\_interfaces.h:



This graph shows which files directly or indirectly include this file:



### Enumerations

- enum [l4vbus\\_iface\\_type\\_t](#)  
*Different sub-interfaces a vbus device may support.*
- enum { [L4VBUS\\_IFACE\\_SHIFT](#) = 26 }

### Functions

- unsigned [l4vbus\\_subinterface](#) (unsigned opcode)  
*Return the ID of the vbus sub-interface.*
- unsigned [l4vbus\\_interface\\_opcode](#) (unsigned opcode)  
*Return the function opcode within the sub-interface of the vbus command.*
- int [l4vbus\\_subinterface\\_supported](#) (l4\_uint32\_t dev\_type, [l4vbus\\_iface\\_type\\_t](#) iface\_type)  
*Check if a vbus device supports a given sub-interface.*

### 15.443.1 Detailed Description

This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

Definition in file [vbus\\_interfaces.h](#).

### 15.443.2 Enumeration Type Documentation

#### 15.443.2.1 anonymous enum

anonymous enum

##### Enumerator

|                    |                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| L4VBUS_IFACE_SHIFT | Sub-interface ID shift. Divides the function opcode sent via IPC into a sub-interface ID and the actual function opcode within the sub-interface. |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 42 of file [vbus\\_interfaces.h](#).

#### 15.443.2.2 l4vbus\_iface\_type\_t

enum [l4vbus\\_iface\\_type\\_t](#)

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see L4VBUS\_IFACE\_SHIFT). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

Definition at line 31 of file [vbus\\_interfaces.h](#).

### 15.443.3 Function Documentation

#### 15.443.3.1 l4vbus\_subinterface\_supported()

```
int l4vbus_subinterface_supported (
 l4_uint32_t dev_type,
 l4vbus_iface_type_t iface_type) [inline]
```

Check if a vbus device supports a given sub-interface.

## Parameters

|                   |                                                              |
|-------------------|--------------------------------------------------------------|
| <i>dev_type</i>   | Device type as reported in <a href="#">l4vbus_device_t</a> . |
| <i>iface_type</i> | Sub-interface type to check for.                             |

## Returns

True if the device supports the sub-interface.

Definition at line 93 of file [vbus\\_interfaces.h](#).

## 15.444 vbus\_interfaces.h

```

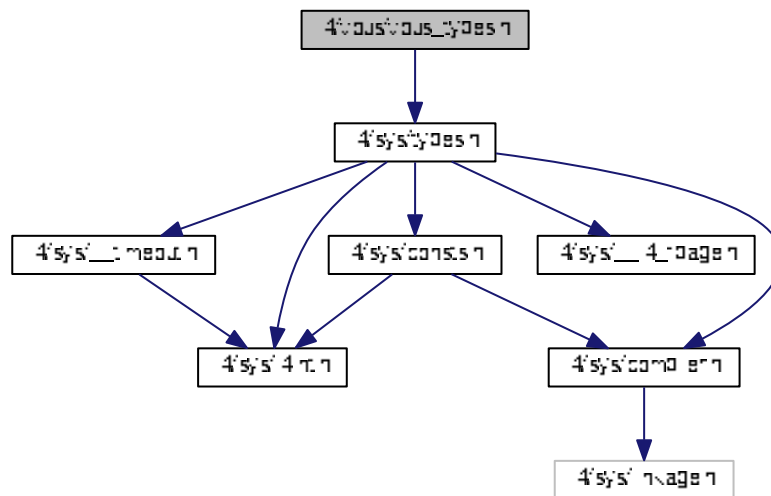
00001 /*
00002 * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003 *
00004 * This file is part of TUD:OS and distributed under the terms of the
00005 * GNU General Public License 2.
00006 * Please see the COPYING-GPL-2 file for details.
00007 */
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016
00031 enum l4vbus_iface_type_t {
00032 L4VBUS_INTERFACE_ICU = 0,
00033 L4VBUS_INTERFACE_GPIO,
00034 L4VBUS_INTERFACE_PCI,
00035 L4VBUS_INTERFACE_PCIDEV,
00036 L4VBUS_INTERFACE_PM,
00037 L4VBUS_INTERFACE_BUS,
00038 L4VBUS_INTERFACE_GENERIC = 0x20
00039 };
00040
00041
00042 enum {
00050 L4VBUS_IFACE_SHIFT = 26
00051 };
00052
00064 L4_INLINE unsigned l4vbus_subinterface(unsigned opcode)
00065 {
00066 return opcode >> L4VBUS_IFACE_SHIFT;
00067 }
00068
00080 L4_INLINE unsigned l4vbus_interface_opcode(unsigned opcode)
00081 {
00082 return opcode & ((1 << L4VBUS_IFACE_SHIFT) - 1);
00083 }
00084
00093 L4_INLINE int l4vbus_subinterface_supported(
00094 l4_uint32_t dev_type,
00095 l4vbus_iface_type_t iface_type)
00096 {
00097 if (iface_type == L4VBUS_INTERFACE_GENERIC)
00098 return 1;
00099 return (dev_type & (1 << iface_type)) ? 1 : 0;
00100 }

```

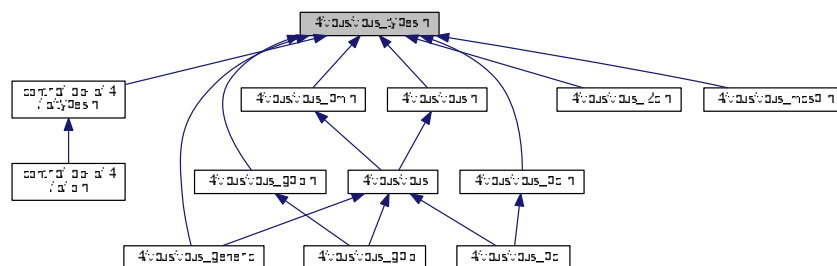
## 15.445 l4/vbus/vbus\_types.h File Reference

This header file contains descriptions of vbus related data types and constants.

```
#include <linux/types.h>
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4vbus_resource_t`  
*Description of a single vbus resource.*
- struct `l4vbus_device_t`  
*Detailed information about a vbus device.*

## Enumerations

- enum `l4vbus_resource_type_t` {  
`L4VBUS_RESOURCE_INVALID` = 0, `L4VBUS_RESOURCE_IRQ`, `L4VBUS_RESOURCE_MEM`, `L4VBUS_RESOURCE_PORT`,  
`L4VBUS_RESOURCE_BUS`, `L4VBUS_RESOURCE_GPIO`, `L4VBUS_RESOURCE_DMA_DOMAIN`, `L4VBUS_RESOURCE_MAX` }
- Description of vbus resource types.*
- enum `l4vbus_device_flags_t` { `L4VBUS_DEVICE_F_CHILDREN` = 0x10 }
- Flags describing device properties, see [l4vbus device t.](#)*

### 15.445.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file [vbus\\_types.h](#).

### 15.445.2 Enumeration Type Documentation

#### 15.445.2.1 l4vbus\_device\_flags\_t

enum [l4vbus\\_device\\_flags\\_t](#)

Flags describing device properties, see [l4vbus\\_device\\_t](#).

##### Enumerator

|                          |                           |
|--------------------------|---------------------------|
| L4VBUS_DEVICE_F_CHILDREN | Device has child devices. |
|--------------------------|---------------------------|

Definition at line 68 of file [vbus\\_types.h](#).

#### 15.445.2.2 l4vbus\_resource\_type\_t

enum [l4vbus\\_resource\\_type\\_t](#)

Description of vbus resource types.

##### Enumerator

|                            |                              |
|----------------------------|------------------------------|
| L4VBUS_RESOURCE_INVALID    | Invalid type.                |
| L4VBUS_RESOURCE_IRQ        | Interrupt resource.          |
| L4VBUS_RESOURCE_MEM        | I/O memory resource.         |
| L4VBUS_RESOURCE_PORT       | I/O port resource (x86 only) |
| L4VBUS_RESOURCE_BUS        | Bus resource.                |
| L4VBUS_RESOURCE_GPIO       | Gpio resource.               |
| L4VBUS_RESOURCE_DMA_DOMAIN | DMA domain.                  |
| L4VBUS_RESOURCE_MAX        | Maximum resource id.         |

Definition at line 39 of file [vbus\\_types.h](#).

## 15.446 vbus\_types.h

00001 /\*

```
00002 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003 * Alexander Warg <warg@os.inf.tu-dresden.de>
00004 * economic rights: Technische Universität Dresden (Germany)
00005 *
00006 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU General Public License 2.
00008 * Please see the COPYING-GPL-2 file for details.
00009 */
00015 #pragma once
00016
00017 #include <l4/sys/types.h>
00018
00019 typedef l4_mword_t l4vbus_device_handle_t;
00020 typedef l4_addr_t l4vbus_paddr_t;
00021
00023 typedef struct {
00025 l4_uint16_t type;
00027 l4_uint16_t flags;
00029 l4vbus_paddr_t start;
00031 l4vbus_paddr_t end;
00033 l4vbus_device_handle_t provider;
00035 l4_uint32_t id;
00036 } l4vbus_resource_t;
00037
00039 enum l4vbus_resource_type_t {
00040 L4VBUS_RESOURCE_INVALID = 0,
00041 L4VBUS_RESOURCE_IRQ,
00042 L4VBUS_RESOURCE_MEM,
00043 L4VBUS_RESOURCE_PORT,
00044 L4VBUS_RESOURCE_BUS,
00045 L4VBUS_RESOURCE_GPIO,
00046 L4VBUS_RESOURCE_DMA_DOMAIN,
00047 L4VBUS_RESOURCE_MAX,
00048 };
00049
00050 enum l4vbus_consts_t {
00051 L4VBUS_DEV_NAME_LEN = 64,
00052 L4VBUS_MAX_DEPTH = 100,
00053 };
00054
00056 typedef struct {
00058 l4_uint32_t type;
00060 char name[L4VBUS_DEV_NAME_LEN];
00062 unsigned num_resources;
00064 unsigned flags;
00065 } l4vbus_device_t;
00066
00068 enum l4vbus_device_flags_t {
00069 L4VBUS_DEVICE_F_CHILDREN = 0x10,
00070 };
```



## Chapter 16

# Example Documentation

### 16.1 examples/clntsrv/client.cc

Client/Server example using C++ infrastructure – Client implementation.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <stdio.h>
#include "shared.h"

int
main()
{
 L4::Cap<Calc> server = L4Re::Env::env()->get_cap<Calc>("calc_server");
 if (!server.is_valid())
 {
 printf("Could not get server capability!\n");
 return 1;
 }

 l4_uint32_t val1 = 8;
 l4_uint32_t val2 = 5;

 printf("Asking for %d - %d\n", val1, val2);

 if (server->sub(val1, val2, &val1))
 {
 printf("Error talking to server\n");
 return 1;
 }
 printf("Result of subtract call: %d\n", val1);
 printf("Asking for -%d\n", val1);
 if (server->neg(val1, &val1))
 {
 printf("Error talking to server\n");
 return 1;
 }
 printf("Result of negate call: %d\n", val1);

 return 0;
}
```

## 16.2 examples/clntsrv/clntsrv.cfg

Sample configuration file for the client/server example.

```
-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Some shortcut for less typing
local ld = L4.default_loader;

-- Channel for the two programs to talk to each other.
local calc_server = ld:new_channel();

-- The server program, getting the channel in server mode.
ld:start({ caps = { calc_server = calc_server:svr() },
 log = { "server", "blue" } },
 "rom/ex_clntsrv-server");

-- The client program, getting the 'calc_server' channel to be able to talk
-- to the server. The client will be started with a green log output.
ld:start({ caps = { calc_server = calc_server },
 log = { "client", "green" } },
 "rom/ex_clntsrv-client");
```

## 16.3 examples/clntsrv/server.cc

Client/Server example using C++ infrastructure – Server implementation.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/util/br_manager>
#include <l4/sys/cxx/ipc_epiface>

#include "shared.h"

static L4Re::Util::Registry_server<L4Re::Util::Br_manager_hooks>
server;

class Calculation_server : public L4::Epiface_t<Calculation_server, Calc>
{
public:
 int op_sub(Calc::Rights, l4_uint32_t a, l4_uint32_t b,
 l4_uint32_t &res)
 {
 res = a - b;
 return 0;
 }

 int op_neg(Calc::Rights, l4_uint32_t a, l4_uint32_t &res)
 {
 res = -a;
 return 0;
 }
};

int
main()
{
 static Calculation_server calc;

 // Register calculation server
 if (!server.registry()->register_obj(&calc, "calc_server").is_valid())
```

```

 {
 printf("Could not register my service, is there a 'calc_server' in the caps table?\n");
 return 1;
 }

 printf("Welcome to the calculation server!\n"
 "I can do substractions and negations.\n");

 // Wait for client requests
 server.loop();

 return 0;
}

```

## 16.4 examples/libs/l4re/c++/mem\_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/mem_alloc>
#include <l4/re/rm>
#include <l4/re/env>
#include <l4/re/dataspace>
#include <l4/re/util/cap_alloc>
#include <l4/sys/err.h>
#include <cstdio>
#include <cstring>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
 void **virt_addr)
{
 int r;
 L4::Cap<L4Re::Dataspace> d;

 /* Allocate a free capability index for our data space */
 d = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
 if (!d.is_valid())
 return -L4_ENOMEM;

 size_in_bytes = l4_trunc_page(size_in_bytes);

 /* Allocate memory via a dataspace */
 if ((r = L4Re::Env::env()->mem_alloc()->alloc(size_in_bytes, d, flags)))
 return r;

 /* Make the dataspace visible in our address space */
 *virt_addr = 0;
 if ((r = L4Re::Env::env()->rm()->attach(virt_addr, size_in_bytes,
 L4Re::Rm::Search_addr,
 L4::Ipc::make_cap_rw(d), 0,
 flags & L4Re::Mem_alloc::Super_pages
 ? L4_SUPERPAGESHIFT :
 L4_PAGESHIFT)))
 return r;

 /* Done, virtual address is in virt_addr */
 return 0;
}

static int free_mem(void *virt_addr)
{
 int r;
 L4::Cap<L4Re::Dataspace> ds;

 /* Detach memory from our address space */
 if ((r = L4Re::Env::env()->rm()->detach(virt_addr, &ds)))
 return r;

 /* Release and return capability slot to allocator */
 L4Re::Util::cap_alloc.free(ds, L4Re::Env::env()->task().cap());
}

```

```

 /* All went ok */
 return 0;
}

int main(void)
{
 void *virt;

 /* Allocate memory: 16k Bytes (usually) */
 if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
 return 1;

 printf("Allocated memory.\n");

 /* Do something with the memory */
 memset(virt, 0x12, 4 * L4_PAGESIZE);

 printf("Touched memory.\n");

 /* Free memory */
 if (free_mem(virt))
 return 2;

 printf("Freed and done. Bye.\n");

 return 0;
}

```

## 16.5 examples/libs/l4re/c++/shared\_ds/ds\_clnt.cc

Sharing memory between applications, client side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/util/cap_alloc> // L4::Cap
#include <l4/re/dataspace> // L4Re::Dataspace
#include <l4/re/rm> // L4::Rm
#include <l4/re/env> // L4::Env
#include <l4/sys/cache.h>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#include "interface.h"

int main()
{
 /*
 * Try to get server interface cap.
 */

 L4::Cap<My_interface> svr = L4Re::Env::env()->
 get_cap<My_interface>("shm");
 if (!svr.is_valid())
 {
 printf("Could not get the server capability\n");
 return 1;
 }

 /*
 * Alloc data space cap slot
 */
 L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.
 alloc<L4Re::Dataspace>();
 if (!ds.is_valid())
 {
 printf("Could not get capability slot!\n");
 return 1;
 }

 /*

```

```

 * Alloc server notifier IRQ cap slot
 */
L4Re::Cap<L4Re::Irq> irq = L4Re::Util::cap_alloc.
 alloc<L4Re::Irq>();
if (!irq.is_valid())
{
 printf("Could not get capability slot!\n");
 return 1;
}

/*
 * Request shared data-space cap.
 */
if (svr->get_shared_buffer(ds, irq)
{
 printf("Could not get shared memory dataspace!\n");
 return 1;
}

/*
 * Attach to arbitrary region
 */
char *addr = 0;
int err = L4Re::Env::env()->rm()->attach(&addr, ds->size(),
 L4Re::Rm::Search_addr,
 L4Re::Ipc::make_cap_rw(ds));

if (err < 0)
{
 printf("Error attaching data space: %s\n", l4sys_errtostr(err));
 return 1;
}

printf("Content: %s\n", addr);

// wait a bit for the demo effect
printf("Sleeping a bit...\n");
sleep(1);

/*
 * Fill in new stuff
 */
memset(addr, 0, ds->size());
char const * const msg = "Hello from client, too!";
printf("Setting new content in shared memory\n");
snprintf(addr, strlen(msg)+1, msg);
l4_cache_clean_data((unsigned long)addr,
 (unsigned long)addr + strlen(msg) + 1);

// notify the server
irq->trigger();

/*
 * Detach region containing addr, result should be Detached_ds (other results
 * only apply if we split regions etc.).
 */
err = L4Re::Env::env()->rm()->detach(addr, 0);
if (err)
 printf("Failed to detach region\n");

/* Free objects and capabilities, just for completeness. */
L4Re::Util::cap_alloc.free(ds, L4Re::This_task);
L4Re::Util::cap_alloc.free(irq, L4Re::This_task);

return 0;
}

```

## 16.6 examples/libs/l4re/c++/shared\_ds/ds\_srv.cc

Sharing memory between applications, server/creator side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

```

```

#include <l4/re/env>
#include <l4/re/namespace>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/dataspace>
#include <l4/cxx/ipc_server>

#include <l4/sys/typeinfo_svr>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#include "interface.h"

class My_server_obj : public L4::Server_object_t<L4::Kobject>
{
private:
 L4::Cap<L4Re::Dataspace> _shm;
 L4::Cap<L4::Irq> _irq;

public:
 explicit My_server_obj(L4::Cap<L4Re::Dataspace> shm,
 L4::Cap<L4::Irq> irq)
 : _shm(shm), _irq(irq)
 {}

 int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int My_server_obj::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
 // we don't care about the original object reference, however
 // we could read out the access rights from the lowest 2 bits
 (void) obj;

 l4_msgtag_t t;
 ios >> t; // extract the tag

 switch (t.label())
 {
 case L4::Meta::Protocol:
 // handle the meta protocol requests, implementing the
 // runtime dynamic type system for L4 objects.
 return L4::Util::handle_meta_request<My_interface>(ios);
 case 0:
 // since we have just one operation we have no opcode dispatch,
 // and just return the data-space and the notifier IRQ capabilities
 ios << _shm << _irq;
 return 0;
 default:
 // every other protocol is not supported.
 return -L4_EBADPROTO;
 }
}

class Shm_observer : public L4::Irq_handler_object
{
private:
 char *_shm;

public:
 explicit Shm_observer(char *shm)
 : _shm(shm)
 {}

 int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int Shm_observer::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
 // We don't care about the original object reference, however
 // we could read out the access rights from the lowest 2 bits
 (void) obj;

 // Since we end up here in this function, we got a 'message' from the IRQ
 // that is bound to us. The 'ios' stream won't contain any valuable info.
 (void) ios;

 printf("Client sent us: %s\n", _shm);

 return 0;
}

static L4Re::Util::Registry_server<> server;

```

```

enum
{
 DS_SIZE = 4 << 12,
};

static char *get_ds(L4::Cap<L4Re::Dataspace> *_ds)
{
 *_ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
 if (!(*_ds).is_valid())
 {
 printf("Dataspace allocation failed.\n");
 return 0;
 }

 int err = L4Re::Env::env()->mem_alloc()->alloc(DS_SIZE, *_ds, 0);
 if (err < 0)
 {
 printf("mem_alloc->alloc() failed.\n");
 L4Re::Util::cap_alloc.free(*_ds);
 return 0;
 }

 /*
 * Attach DS to local address space
 */
 char *_addr = 0;
 err = L4Re::Env::env()->rm()->attach(&_amp;_addr, (*_ds)->size(),
 L4Re::Rm::Search_addr,
 L4::Ipc::make_cap_rw(*_ds));

 if (err < 0)
 {
 printf("Error attaching data space: %s\n", l4sys_errtostr(err));
 L4Re::Util::cap_alloc.free(*_ds);
 return 0;
 }

 /*
 * Success! Write something to DS.
 */
 printf("Attached DS\n");
 static char const * const msg = "[DS] Hello from server!";
 snprintf(_addr, strlen(msg) + 1, msg);

 return _addr;
}

int main()
{
 L4::Cap<L4Re::Dataspace> ds;
 char *addr;

 if (!(addr = get_ds(&ds)))
 return 2;

 // First the IRQ handler, because we need it in the My_server_obj object
 Shm_observer observer(addr);

 // Registering the observer as an IRQ handler, this allocates an
 // IRQ object using the factory of our server.
 L4::Cap<L4::Irq> irq = server.registry()->register_irq_obj(&observer);

 // Now the initial server object shared with the client via our parent.
 // it provides the data-space and the IRQ capabilities to a client.
 My_server_obj server_obj(ds, irq);

 // Registering the server object to the capability 'shm' in our the L4Re::Env.
 // This capability must be provided by the parent. (see the shared_ds.lua)
 server.registry()->register_obj(&server_obj, "shm");

 // Run our server loop.
 server.loop();
 return 0;
}

```

## 16.7 examples/libs/l4re/c++/shared\_ds/shared\_ds.cfg

Sharing memory between applications, configuration file.

```
-- Include L4 functionality
local L4 = require("L4");

-- Create a channel from the client to the server
local channel = L4.default_loader:new_channel();

-- Start the server, giving the channel with full server rights.
-- The server will have a yellow log output.
L4.default_loader:start(
{
 caps = { shm = channel:svr() },
 log = { "server", "yellow" }
},
"rom/ex_l4re_ds_srv"
);

-- Start the client, giving it the channel with read only rights. The
-- log output will be green.
L4.default_loader:start(
{
 caps = { shm = channel },
 log = { "client", "green" },
 l4re_dbg = L4.Dbg.Warn
},
"rom/ex_l4re_ds_clnt"
);
```

## 16.8 examples/libs/l4re/c/ma+rm.c

Coarse grained memory allocation, in C.

```
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4re/c/mem_alloc.h>
#include <l4re/c/rm.h>
#include <l4re/c/util/cap_alloc.h>
#include <l4/sys/err.h>
#include <stdio.h>
#include <string.h>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
 void **virt_addr)
{
 int r;
 l4re_ds_t ds;

 /* Allocate a free capability index for our data space */
 ds = l4re_util_cap_alloc();
 if (l4_is_invalid_cap(ds))
 return -L4_ENOMEM;

 size_in_bytes = l4_trunc_page(size_in_bytes);

 /* Allocate memory via a dataspace */
 if ((r = l4re_ma_alloc(size_in_bytes, ds, flags))
 return r;

 /* Make the dataspace visible in our address space */
 *virt_addr = 0;
 if ((r = l4re_rm_attach(virt_addr, size_in_bytes,
 L4RE_RM_SEARCH_ADDR, ds, 0,
 flags & L4RE_MA_SUPER_PAGES
 ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
 {
 /* Free dataspace again */
 l4re_util_cap_free_um(ds);
 return r;
 }

 /* Done, virtual address is in virt_addr */
 return 0;
}
```



```

static int free_mem(void *virt_addr)
{
 int r;
 l4re_ds_t ds;

 /* Detach memory from our address space */
 if ((r = l4re_rm_detach_ds(virt_addr, &ds))
 return r;

 /* Free memory at our memory allocator */
 l4re_util_cap_free_um(ds);

 /* All went ok */
 return 0;
}

int main(void)
{
 void *virt;

 /* Allocate memory: 16k Bytes (usually) */
 if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
 return 1;

 printf("Allocated memory.\n");

 /* Do something with the memory */
 memset(virt, 0x12, 4 * L4_PAGESIZE);

 printf("Touched memory.\n");

 /* Free memory */
 if (free_mem(virt))
 return 2;

 printf("Freed and done. Bye.\n");

 return 0;
}

```

## 16.9 examples/libs/l4re/streammap/client.cc

Client/Server example showing how to map a page to another task – Client implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/cxx/ipc_stream>

#include <stdio.h>

#include "shared.h"

static int
func_smap_call(L4::Cap<void> const &server)
{
 L4::Ipc::Iostream s(l4_utcb());
 l4_addr_t addr = 0;
 int err;

 if ((err = L4Re::Env::env()->rm()->reserve_area(&addr,
 L4_PAGESIZE,
 L4Re::Rm::Search_addr)))
 {
 printf("The reservation of one page within our virtual memory failed with %d\n", err);
 return 1;
 }
}

```

```

s << L4::Opcode(Mapper::Do_map)
 << (l4_addr_t)addr;
s << L4::Ipc::Rcv_fpage::mem((l4_addr_t)addr, L4_PAGESHIFT, 0);
int r = l4_error(s.call(server.cap(), Mapper::Protocol));
if (r)
 return r; // failure

printf("String sent by server: %s\n", (char *)addr);

return 0; // ok
}

int
main()
{
 L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("smap");
 if (!server.is_valid())
 {
 printf("Could not get capability slot!\n");
 return 1;
 }

 printf("Asking for page from server\n");

 if (func_smap_call(server))
 {
 printf("Error talking to server\n");
 return 1;
 }
 printf("It worked!\n");

 L4Re::Util::cap_alloc.free(server, L4Re::This_task);

 return 0;
}

```

## 16.10 examples/libs/l4re/streammap/server.cc

Client/Server example showing how to map a page to another task – Server implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/ipc_server>

#include "shared.h"

static char page_to_map[L4_PAGESIZE] __attribute__((aligned(
 L4_PAGESIZE)));

static L4Re::Util::Registry_server<> server;

class Smap_server : public L4::Server_object_t<Mapper>
{
public:
 int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int
Smap_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)
{
 l4_msgtag_t t;
 ios >> t;

 // We're only talking the Map_example protocol

```

```

if (t.label() != Mapper::Protocol)
 return -L4_EBADPROTO;

L4::Opcode opcode;
ios >> opcode;

switch (opcode)
{
 case Mapper::Do_map:
 l4_addr_t snd_base;
 ios >> snd_base;
 // put something into the page to read it out at the other side
 snprintf(page_to_map, sizeof(page_to_map), "Hello from the server!");
 printf("Sending to client\n");
 // send page
 ios << L4::Ipc::Snd_fpage::mem((l4_addr_t)page_to_map,
 L4_PAGESHIFT,
 L4_FPAGE_RO, snd_base);

 return L4_EOK;
 default:
 return -L4_ENOSYS;
}
}

int
main()
{
 static Smap_server smap;

 // Register server
 if (!server.registry()->register_obj(&smap, "smap").is_valid())
 {
 printf("Could not register my service, read-only namespace?\n");
 return 1;
 }

 printf("Welcome to the memory map example server!\n");

 // Wait for client requests
 server.loop();

 return 0;
}

```

## 16.11 examples/libs/l4re/streammap/streammap.cfg

Sample configuration file for the client/server map example.

```

-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Channel for the communication between the server and the client.
local smap_channel = L4.default_loader:new_channel();

-- The server program, using the 'smap' channel in server
-- Mode. The log prefix will be 'server', colored yellow.
L4.default_loader:start({ caps = { smap = smap_channel:svr() },
 log = { "server", "yellow" }},
 "rom/ex_smap-server");

-- The client program.
-- It is given the 'smap' channel to be able to talk to the server.
-- The log prefix will be 'client', colored green.
L4.default_loader:start({ caps = { smap = smap_channel },
 log = { "client", "green" }},
 "rom/ex_smap-client");

```

## 16.12 examples/libs/libirq/async\_isr.c

libirq usage example using asynchronous ISR handler functionality.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to use the libirq.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>

#include <stdio.h>

enum { IRQ_NO = 17 };

static void isr_handler(void *data)
{
 (void)data;
 printf("Got IRQ %d\n", IRQ_NO);
}

int main(void)
{
 const int seconds = 5;
 l4irq_t *irqdesc;

 if (!(irqdesc = l4irq_request(IRQ_NO, isr_handler, 0, 0xff, 0)))
 {
 printf("Requesting IRQ %d failed\n", IRQ_NO);
 return 1;
 }

 printf("Attached to key IRQ %d\nPress keys now, will terminate in %d seconds\n",
 IRQ_NO, seconds);

 l4_sleep(seconds * 1000);

 if (l4irq_release(irqdesc))
 {
 printf("Failed to release IRQ\n");
 return 1;
 }

 printf("Bye\n");
 return 0;
}

```

## 16.13 examples/libs/libirq/loop.c

libirq usage example using a self-created thread.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>
#include <stdio.h>
#include <pthread.h>

enum { IRQ_NO = 17 };

static void isr_handler(void)
{
 printf("Got IRQ %d\n", IRQ_NO);
}

static void *isr_thread(void *data)
{
 l4irq_t *irq;
 (void)data;

```

```

 if (!(irq = l4irq_attach(IRQ_NO)))
 return NULL;

 while (1)
 {
 if (l4irq_wait(irq))
 continue;
 isr_handler();
 }

 return NULL;
}

int main(void)
{
 pthread_t thread;

 if (pthread_create(&thread, NULL, isr_thread, NULL))
 return 1;

 l4_sleep_forever();
 return 0;
}

```

## 16.14 examples/libs/shmc/prodcons.c

Simple shared memory example.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

/*
 * This example uses shared memory between two threads, one producer, one
 * consumer.
 */

#include <l4/shmc/shmc.h>

#include <l4/util/util.h>

#include <stdio.h>
#include <string.h>
#include <pthread-l4.h>

#include <l4/sys/thread.h>

// a small helper
#define CHK(func) if (func) { printf("failure: %d\n", __LINE__); return (void *)-1; }

static const char some_data[] = "Hi consumer!";

static void *thread_producer(void *d)
{
 (void)d;
 l4shmc_chunk_t p_one;
 l4shmc_signal_t s_one, s_done;
 l4shmc_area_t shmarea;

 // attach this thread to the shm object
 CHK(l4shmc_attach("testshm", &shmarea));

 // add a chunk
 CHK(l4shmc_add_chunk(&shmarea, "one", 1024, &p_one));

 // add a signal
 CHK(l4shmc_add_signal(&shmarea, "prod", &s_one));

 CHK(l4shmc_attach_signal_to(&shmarea, "done",
 pthread_l4_cap(pthread_self()), 10000, &s_done));

 // connect chunk and signal
 CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));
}

```

```

printf("PRODUCER: ready\n");

while (1)
{
 while (l4shmc_chunk_try_to_take(&p_one))
 printf("Uh, should not happen!\n"); //l4_thread_yield();

 memcpy(l4shmc_chunk_ptr(&p_one), some_data, sizeof(some_data));

 CHK(l4shmc_chunk_ready_sig(&p_one, sizeof(some_data)));

 printf("PRODUCER: Sent data\n");

 CHK(l4shmc_wait_signal(&s_done));
}

l4_sleep_forever();
return NULL;
}

static void *thread_consume(void *d)
{
 (void)d;
 l4shmc_area_t shmarea;
 l4shmc_chunk_t p_one;
 l4shmc_signal_t s_one, s_done;

 // attach to shared memory area
 CHK(l4shmc_attach("testshm", &shmarea));

 // get chunk 'one'
 CHK(l4shmc_get_chunk(&shmarea, "one", &p_one));

 // add a signal
 CHK(l4shmc_add_signal(&shmarea, "done", &s_done));

 // attach signal to this thread
 CHK(l4shmc_attach_signal_to(&shmarea, "prod",
 pthread_l4_cap(pthread_self(), 10000, &s_one));

 // connect chunk and signal
 CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

 while (1)
 {
 CHK(l4shmc_wait_chunk(&p_one));

 printf("CONSUMER: Received from chunk one: %s\n",
 (char *)l4shmc_chunk_ptr(&p_one));
 memset(l4shmc_chunk_ptr(&p_one), 0, l4shmc_chunk_size(&p_one));

 CHK(l4shmc_chunk_consumed(&p_one));
 CHK(l4shmc_trigger(&s_done));
 }

 return NULL;
}

int main(void)
{
 pthread_t one, two;

 // create new shared memory area, 8K in size
 if (l4shmc_create("testshm", 8192))
 return 1;

 // create two threads, one for producer, one for consumer
 pthread_create(&one, 0, thread_producer, 0);
 pthread_create(&two, 0, thread_consume, 0);

 // now sleep, the two threads are doing the work
 l4_sleep_forever();

 return 0;
}

```

## 16.15 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>,
 * Björn Döbel <doebel@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Example to show syscall tracing.
 */
#if defined(ARCH_x86) || defined(ARCH_amd64)
// MEASURE only works on x86/amd64
// #define MEASURE
#endif

#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/sys/debugger.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/* Architecture specifics */
#if defined(ARCH_x86) || defined(ARCH_amd64)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{
 #if defined(ARCH_x86)
 return exc->err & 4;
 #else
 return exc->err == 1;
 #endif
}

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
 printf("PC=%08lx SP=%08lx Err=%08lx Trap=%lx, %s syscall, SC-Nr: %lx\n",
 l4_utcb_exc_pc(exc), exc->sp, exc->err,
 exc->trapno, is_alien_after_call(exc) ? " after" : "before",
 exc->err >> 3);
}

#elif defined(ARCH_arm)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{
 return exc->err & 0x40; } // TODO: Should change this to (1 << 16)

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
 printf("PC=%08lx SP=%08lx ULR=%08lx CPSR=%08lx Err=%lx/%lx, %s syscall\n",
 l4_utcb_exc_pc(exc), exc->sp, exc->ulr, exc->cpsr,
 exc->err, exc->err >> 26,
 is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_arm64)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{
 return exc->err & (1ul << 16); }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
 printf("PC=%08lx SP=%08lx PSTATE=%08lx Err=%lx/%lx, %s syscall\n",
 l4_utcb_exc_pc(exc), exc->sp, exc->pstate,
 exc->err, exc->err >> 26,
 is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_mips)

static int

```

```

is_alien_after_call(l4_exc_regs_t const *exc)
{ return 0; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
 printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
 l4_utcb_exc_pc(exc), exc->sp, exc->cause,
 is_alien_after_call(exc) ? " after" : "before");
}

#else

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 1; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
 printf("PC=%08lx SP=%08lx, %s syscall\n",
 l4_utcb_exc_pc(exc), exc->sp,
 is_alien_after_call(exc) ? " after" : "before");
}

#endif

/* Measurement mode specifics.
 *
 * In measurement mode the code is less verbose and uses RDTSC for alien exception
 * performance measurement.
 */
#ifdef MEASURE

#include <l4/util/rdtsc.h>

static inline void
calibrate_timer(void)
{
 l4_calibrate_tsc(l4re_kip());
}

static inline void
print_timediff(l4_cpu_time_t start)
{
 e = l4_rdtsc();
 printf("time %lld\n", l4_tsc_to_ns(e - start));
}

static inline void
alien_sleep(void)
{
 l4_sleep(0);
}

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
 if (0)
 _print_exc_state(exc);
}

#else

static inline void
calibrate_timer(void)
{
}

static inline void
print_timediff(l4_cpu_time_t start)
{
 (void)start;
}

static inline l4_cpu_time_t
l4_rdtsc(void)
{
 return 0;
}

static inline void
alien_sleep(void)
{
 l4_sleep(1000);
}

```



```

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
 _print_exc_state(exc);
}

#endif

static char alien_thread_stack[8 << 10];
static l4_cap_idx_t alien;

static void alien_thread(void)
{
 while (1)
 {
 l4_ipc_call(0x1234 << L4_CAP_SHIFT, l4_utcb(),
 l4_msgtag(0, 0, 0, 0), L4_IPC_NEVER);
 alien_sleep();
 }
}

int main(void)
{
 l4_msgtag_t tag;
 l4_cpu_time_t s;
 l4_utcb_t *u = l4_utcb();
 l4_exc_regs_t exc;
 l4_umword_t mr0, mr1;

 printf("Alien feature testing\n");

 l4_debugger_set_object_name(l4re_env()->main_thread, "alientest");

 /* Start alien thread */
 if (l4_is_invalid_cap(alien = l4re_util_cap_alloc()))
 return 1;

 l4_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));

 tag = l4_factory_create_thread(l4re_env()->factory, alien);
 if (l4_error(tag))
 return 2;

 l4_debugger_set_object_name(alien, "alienth");

 l4_addr_t kumem;
 if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP,
 l4re_env()->rm))
 return 3;

 l4_thread_control_start();
 l4_thread_control_pager(l4re_env()->main_thread);
 l4_thread_control_exc_handler(l4re_env()->main_thread);
 l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
 l4_thread_control_alien(1);
 tag = l4_thread_control_commit(alien);
 if (l4_error(tag))
 return 4;

 tag = l4_thread_ex_regs(alien,
 (l4_umword_t)alien_thread,
 (l4_umword_t)alien_thread_stack + sizeof(alien_thread_stack),
 0);

 if (l4_error(tag))
 return 5;

 l4_sched_param_t sp = l4_sched_param(1, 0);
 tag = l4_scheduler_run_thread(l4re_env()->scheduler, alien, &sp);
 if (l4_error(tag))
 return 6;

 calibrate_timer();

 /* Pager/Exception loop */
 if (l4_msgtag_has_error(tag = l4_ipc_receive(alien, u,
 L4_IPC_NEVER)))
 {
 printf("l4_ipc_receive failed");
 return 7;
 }

 memcpy(&exc, l4_utcb_exc(), sizeof(exc));
 mr0 = l4_utcb_mr()->mr[0];
 mr1 = l4_utcb_mr()->mr[1];

 for (;;)

```

```

{
 s = l4_rdtsc();

 if (l4_msgtag_is_exception(tag))
 {
 print_exc_state(&exc);
 tag = l4_msgtag(is_alien_after_call(&exc)
 ? 0 : L4_PROTO_ALLOW_SYSCALL,
 L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
 }
 else
 printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);

 memcpy(l4_utcb_exc(), &exc, sizeof(exc));

 /* Reply and wait */
 if (l4_msgtag_has_error(tag = l4_ipc_call(alien, u, tag,
 L4_IPC_NEVER)))
 {
 printf("l4_ipc_call failed\n");
 return 8;
 }
 memcpy(&exc, l4_utcb_exc(), sizeof(exc));
 mr0 = l4_utcb_mr()->mr[0];
 mr1 = l4_utcb_mr()->mr[1];
 print_timediff(s);
}

return 0;
}

```

## 16.16 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

```

vim:se ft=lua:

local L4 = require("L4");

L4.default_loader:start({}, "rom/ex_ipc1");

```

## 16.17 examples/sys/ipc/ipc\_example.c

This example shows how two threads can exchange data using the [L4](#) IPC mechanism. One thread is sending an integer to the other thread which is returning the square of the integer. Both values are printed.

```

/*
 * (c) 2008-2009 Author(s)
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>

#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>

static pthread_t t2;

/* Thread1 is the initiator thread, i.e. it initiates the IPC calls. In
 * other words, it takes the client role. It uses L4 IPC mechanisms to send
 * an integer value to thread2 and received a calculation result back. */
static void *thread1_fn(void *arg)
{
 l4_msgtag_t tag;
 int ipc_error;
 unsigned long value = 1;

```

```

(void) arg;

while (1)
{
 printf("Sending: %ld\n", value);

 /* Store the value which we want to have squared in the first message
 * register of our UTCB. */
 l4_utcb_mr()->mr[0] = value;

 /* To an L4 IPC call, i.e. send a message to thread2 and wait for a
 * reply from thread2. The '1' in the msgtag denotes that we want to
 * transfer one word of our message registers (i.e. MR0). No timeout. */
 tag = l4_ipc_call(pthread_l4_cap(t2), l4_utcb(),
 l4_msgtag(0, 1, 0, 0), L4_IPC_NEVER);
 /* Check for IPC error, if yes, print out the IPC error code, if not,
 * print the received result. */
 ipc_error = l4_ipc_error(tag, l4_utcb());
 if (ipc_error)
 fprintf(stderr, "thread1: IPC error: %x\n", ipc_error);
 else
 printf("Received: %ld\n", l4_utcb_mr()->mr[0]);

 /* Wait some time and increment our value. */
 sleep(1);
 value++;
}
return NULL;
}

/* Thread2 is in the server role, i.e. it waits for requests from others and
 * sends back the calculation results. */
static void *thread2_fn(void *arg)
{
 l4_msgtag_t tag;
 l4_umword_t label;
 int ipc_error;
 (void) arg;

 /* Wait for requests from any thread. No timeout, i.e. wait forever. */
 tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
 while (1)
 {
 /* Check if we had any IPC failure, if yes, print the error code
 * and just wait again. */
 ipc_error = l4_ipc_error(tag, l4_utcb());
 if (ipc_error)
 {
 fprintf(stderr, "thread2: IPC error: %x\n", ipc_error);
 tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
 continue;
 }

 /* So, the IPC was ok, now take the value out of message register 0
 * of the UTCB and store the square of it back to it. */
 l4_utcb_mr()->mr[0] = l4_utcb_mr()->mr[0] *
 l4_utcb_mr()->mr[0];

 /* Send the reply and wait again for new messages.
 * The '1' in the msgtag indicated that we want to transfer 1 word in
 * the message registers (i.e. MR0) */
 tag = l4_ipc_reply_and_wait(l4_utcb(),
 l4_msgtag(0, 1, 0, 0),
 &label, L4_IPC_NEVER);
 }
 return NULL;
}

int main(void)
{
 // We will have two threads, one is already running the main function, the
 // other (thread2) will be created using pthread_create.

 if (pthread_create(&t2, NULL, thread2_fn, NULL))
 {
 fprintf(stderr, "Thread creation failed\n");
 return 1;
 }

 // Just run thread1 in the main thread
 thread1_fn(NULL);
 return 0;
}

```

## 16.18 examples/sys/isr/main.c

Example of an interrupt service routine.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>,
 * Björn Döbel <doebel@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to connect to an interrupt, receive interrupt
 * events and detach again. As the interrupt source we'll use the virtual
 * key interrupt. The interrupt number of the virtual key interrupt can be
 * found in the kernel info page.
 */

#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/namespace.h>
#include <l4/sys/utcb.h>
#include <l4/sys/irq.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>

#include <stdio.h>

int main(void)
{
 int irqno = 1;
 l4_cap_idx_t irqcap, icucap;
 l4_msgtag_t tag;
 int err;
 icucap = l4re_env_get_cap("icu");

 /* Get a free capability slot for the ICU capability */
 if (l4_is_invalid_cap(icucap))
 {
 printf("Did not find an ICU\n");
 return 1;
 }

 /* Get another free capability slot for the corresponding IRQ object */
 if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))
 return 1;
 /* Create IRQ object */
 if (l4_error(tag = l4_factory_create_irq(l4re_global_env->
 factory, irqcap)))
 {
 printf("Could not create IRQ object: %lx\n", l4_error(tag));
 return 1;
 }

 /*
 * Bind the recently allocated IRQ object to the IRQ number irqno
 * as provided by the ICU.
 */
 if (l4_error(l4_icu_bind(icucap, irqno, irqcap)))
 {
 printf("Binding IRQ%d to the ICU failed\n", irqno);
 return 1;
 }

 /* Bind ourselves to the IRQ */
 tag = l4_rcv_ep_bind_thread(irqcap, l4re_env()->main_thread, 0xDEAD);
 if ((err = l4_error(tag)))
 {
 printf("Error binding to IRQ %d: %d\n", irqno, err);
 return 1;
 }

 printf("Attached to key IRQ %d\nPress keys now, Shift-Q to exit\n", irqno);

 /* IRQ receive loop */
 while (1)
 {
 unsigned long label = 0;
 /* Wait for the interrupt to happen */
 tag = l4_irq_receive(irqcap, L4_IPC_NEVER);
 if ((err = l4_ipc_error(tag, l4_utcb())))

```

```

 printf("Error on IRQ receive: %d\n", err);
 else
 {
 /* Process the interrupt -- may do a 'break' */
 printf("Got IRQ with label 0x%lX\n", label);
 }
}

/* We're done, detach from the interrupt. */
tag = l4_irq_detach(irqcap);
if ((err = l4_error(tag)))
 printf("Error detach from IRQ: %d\n", err);

return 0;
}

```

## 16.19 examples/sys/migrate/thread\_migrate.cc

Thread migration example.

```

/*
 * (c) 2008-2009 Author(s)
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/scheduler>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

enum { NR_THREADS = 12 };
static L4::Cap<L4::Thread> threads[NR_THREADS];
static l4_umword_t cpu_map, cpu_nrs;

/* Function for the threads. The content is not really relevant, so lets
 * just sleep around a bit. */
static void *thread_fn(void *)
{
 while (1)
 sleep(1);

 return 0;
}

/* Check how many CPUs we have available.
 */
static int check_cpus(void)
{
 l4_sched_cpu_set_t cs = l4_sched_cpu_set(0, 0);

 if (l4_error(L4Re::Env::env()->scheduler()->info(&cpu_nrs, &cs)) < 0)
 return 1;

 cpu_map = cs.map;

 printf("%ld maximal supported CPUs.\n", cpu_nrs);
 if (cpu_nrs >= L4_MWORD_BITS)
 {
 printf("Will only handle %ld CPUs.\n", cpu_nrs);
 cpu_nrs = L4_MWORD_BITS;
 }
 else if (cpu_nrs == 1)
 printf("Only found 1 CPU.\n");

 return cpu_nrs < 2;
}

/* Create a couple of threads and store their capabilities in an array */
static int create_threads(void)
{
 unsigned i;

```

```

for (i = 0; i < NR_THREADS; ++i)
{
 pthread_t t;

 if (pthread_create(&t, NULL, thread_fn, NULL))
 return 1;

 threads[i] = L4::Cap<L4::Thread>(pthread_l4_cap(t));
}
printf("Created %d threads.\n", NR_THREADS);
return 0;
}

/* Helper function to get the next CPU */
static unsigned get_next_cpu(unsigned c)
{
 unsigned x = c;
 for (;;)
 {
 x = (x + 1) % cpu_nrs;
 if (L4Re::Env::env()->scheduler()->is_online(x))
 return x;
 if (x == c)
 return c;
 }
}

/* Function that shuffles the threads on the available CPUs */
static void shuffle(void)
{
 unsigned start = 0;
 while (1)
 {
 unsigned t;
 unsigned c = start;
 for (t = 0; t < NR_THREADS; ++t)
 {
 l4_sched_param_t sp = l4_sched_param(20);
 c = get_next_cpu(c);
 sp.affinity = l4_sched_cpu_set(c, 0);
 if (l4_error(L4Re::Env::env()->scheduler()->run_thread(threads[t], sp)))
 printf("Error migrating thread%02d to CPU%02d\n", t, c);
 printf("Migrated Thread%02d -> CPU%02d\n", t, c);
 }

 start++;
 if (start == cpu_nrs)
 start = 0;
 sleep(1);
 }
}

int main(void)
{
 if (check_cpus())
 return 1;

 if (create_threads())
 return 1;

 shuffle();

 return 0;
}

```

## 16.20 examples/sys/migrate/thread\_migrate.cfg

Sample configuration file for the thread migration example.

```

-- vim:set ft=lua:

local L4 = require("L4");

-- The log prefix will be 'migrate', colored green.
L4.default_loader:start({ log = { "migrate", "green" } },
 "rom/ex_thread_migrate");

```

## 16.21 examples/sys/singlestep/main.c

This example shows how a thread can be single stepped on the x86 architecture.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>,
 * Björn Döbel <doebel@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Single stepping example for the x86-32 architecture.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/factory.h>
#include <l4/sys/thread.h>
#include <l4/sys/utcb.h>
#include <l4/sys/kdebug.h>

#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char thread_stack[8 << 10];

static void thread_func(void)
{
 while (1)
 {
 unsigned long d = 0;

 /* Enable single stepping */
 asm volatile("pushf; pop %0; or $256,%0; push %0; popf\n"
 : "=r" (d) : "r" (d));

 /* Some instructions */
 asm volatile("nop");
 asm volatile("nop");
 asm volatile("nop");
 asm volatile("mov $0x12345000, %%edx" : : : "edx"); // a non-existent cap
 asm volatile("int $0x30\n");
 asm volatile("nop");
 asm volatile("nop");
 asm volatile("nop");

 /* Disabled single stepping */
 asm volatile("pushf; pop %0; and $~256,%0; push %0; popf\n"
 : "=r" (d) : "r" (d));

 /* You won't see those */
 asm volatile("nop");
 asm volatile("nop");
 asm volatile("nop");
 }
}

int main(void)
{
 l4_msgtag_t tag;
 int ipc_stat = 0;
 l4_cap_idx_t th = l4re_util_cap_alloc();
 l4_exc_regs_t exc;
 l4_umword_t mr0, mr1;
 l4_utcb_t *u = l4_utcb();

 printf("Singlestep testing\n");

 if (l4_is_invalid_cap(th))
 return 1;

 l4_touch_rw(thread_stack, sizeof(thread_stack));
 l4_touch_ro(thread_func, 1);

 tag = l4_factory_create_thread(l4re_env()->factory, th);
 if (l4_error(tag))

```

```

 return 1;

l4_thread_control_start();
l4_thread_control_pager(l4re_env()->main_thread);
l4_thread_control_exc_handler(l4re_env()->main_thread);
l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
 L4RE_THIS_TASK_CAP);
l4_thread_control_alien(1);
tag = l4_thread_control_commit(th);
if (l4_error(tag))
 return 2;

tag = l4_thread_ex_regs(th, (l4_umword_t)thread_func,
 (l4_umword_t)thread_stack + sizeof(thread_stack),
 0);

if (l4_error(tag))
 return 3;

l4_sched_param_t sp = l4_sched_param(1, 0);
tag = l4_scheduler_run_thread(l4re_env()->scheduler, th, &sp);
if (l4_error(tag))
 return 4;

/* Pager/Exception loop */
if (l4_msgtag_has_error(tag = l4_ipc_receive(th, u,
 L4_IPC_NEVER)))
{
 printf("l4_ipc_receive failed");
 return 5;
}
memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr()->mr[0];
mr1 = l4_utcb_mr()->mr[1];

for (;;)
{
 if (l4_msgtag_is_exception(tag))
 {
 printf("PC = %08lx Trap = %08lx Err = %08lx, SP = %08lx SC-Nr: %lx\n",
 l4_utcb_exc_pc(&exc), exc.trapno, exc.err,
 exc.sp, exc.err >> 3);
 if (exc.err >> 3)
 {
 if (!(exc.err & 4))
 {
 tag = l4_msgtag(L4_PROTO_ALLOW_SYSCALL,
 L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

 if (ipc_stat)
 enter_kdebug("Should not be 1");
 }
 else
 {
 tag = l4_msgtag(L4_PROTO_NONE,
 L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

 if (!ipc_stat)
 enter_kdebug("Should not be 0");
 }
 ipc_stat = !ipc_stat;
 }
 l4_sleep(100);
 }
 else
 printf("Umm, non-handled request: %ld, %08lx %08lx\n",
 l4_msgtag_label(tag), mr0, mr1);

 memcpy(l4_utcb_exc(), &exc, sizeof(exc));

 /* Reply and wait */
 if (l4_msgtag_has_error(tag = l4_ipc_call(th, u, tag,
 L4_IPC_NEVER)))
 {
 printf("l4_ipc_call failed\n");
 return 5;
 }
 memcpy(&exc, l4_utcb_exc(), sizeof(exc));
 mr0 = l4_utcb_mr()->mr[0];
 mr1 = l4_utcb_mr()->mr[1];
}

return 0;
}

```



## 16.22 examples/sys/start-with-exc/main.c

This example shows how to start a newly created thread with a defined set of CPU registers.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>,
 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Start a thread with an exception reply. This example does only work on
 * the x86-32 and ARM architectures.
 */

#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/ipc.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>

/* Stack for the thread to be created. 8kB are enough. */
static char thread_stack[8 << 10];

/* The thread to be created. For illustration it will print out its
 * register set.
 */
static void L4_STICKY(thread_func(l4_umword_t *d))
{
 while (1)
 {
 printf("hey, I'm a thread\n");
 printf("got register values: %ld %ld %ld %ld %ld %ld %ld %ld\n",
 d[7], d[6], d[5], d[4], d[2], d[1], d[0]);
 l4_sleep(800);
 }
}

/* Startup trick for this example. Put all the CPU registers on the stack so
 * that the C function above can get it on the stack. */
asm(
 ".global thread\n\t"
 "thread:\n\t"
 "#ifdef ARCH_x86\n\t"
 " pusha\n\t"
 " push %esp\n\t"
 " call thread_func\n\t"
 "#endif\n\t"
 "#ifdef ARCH_arm\n\t"
 " push {r0-r7}\n\t"
 " mov r0, sp\n\t"
 " bl thread_func\n\t"
 "#endif\n\t"
);
extern void thread(void);

/* Our main function */
int main(void)
{
 /* Get a capability slot for our new thread. */
 l4_cap_idx_t t1 = l4re_util_cap_alloc();
 l4_utcb_t *u = l4_utcb();
 l4_exc_regs_t *e = l4_utcb_exc_u(u);
 l4_msgtag_t tag;
 int err;

 printf("Example showing how to start a thread with an exception.\n");
 /* We do not want to implement a pager here, take the shortcut. */
 printf("Make sure to start this program with ldr-flags=eager_map\n");

 if (l4_is_invalid_cap(t1))
 return 1;

```

```

/* Create the thread using our default factory */
tag = l4_factory_create_thread(l4re_env()->factory, t1);
if (l4_error(tag))
 return 1;

/* Setup the thread by setting the pager and task. */
l4_thread_control_start();
l4_thread_control_pager(l4re_env()->main_thread);
l4_thread_control_exc_handler(l4re_env()->main_thread);
l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
 L4RE_THIS_TASK_CAP);
tag = l4_thread_control_commit(t1);
if (l4_error(tag))
 return 2;

/* Start the thread by finally setting instruction and stack pointer */
tag = l4_thread_ex_regs(t1,
 (l4_umword_t)thread,
 (l4_umword_t)thread_stack + sizeof(thread_stack),
 L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
if (l4_error(tag))
 return 3;

l4_sched_param_t sp = l4_sched_param(1, 0);
tag = l4_scheduler_run_thread(l4re_env()->scheduler, t1, &sp);
if (l4_error(tag))
 return 4;

/* Receive initial exception from just started thread */
tag = l4_ipc_receive(t1, u, L4_IPC_NEVER);
if ((err = l4_ipc_error(tag, u)))
{
 printf("Umm, ipc error: %x\n", err);
 return 1;
}
/* We expect an exception IPC */
if (!l4_msgtag_is_exception(tag))
{
 printf("PF?: %lx %lx (not prepared to handle this) %ld\n",
 l4_utcb_mr_u(u)->mr[0], l4_utcb_mr_u(u)->mr[1], l4_msgtag_label(tag));
 return 1;
}

/* Fill out the complete register set of the new thread */
e->sp = (l4_umword_t)(thread_stack + sizeof(thread_stack));
#ifdef ARCH_x86
e->ip = (l4_umword_t)thread;
e->edi = 0;
e->esi = 1;
e->ebp = 2;
e->ebx = 4;
e->edx = 5;
e->ecx = 6;
e->eax = 7;
#endif
#ifdef ARCH_arm
e->pc = (l4_umword_t)thread;
e->r[0] = 0;
e->r[1] = 1;
e->r[2] = 2;
e->r[3] = 3;
e->r[4] = 4;
e->r[5] = 5;
e->r[6] = 6;
e->r[7] = 7;
#endif
/* Send a complete exception */
tag = l4_msgtag(0, L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

/* Send reply and start the thread with the defined CPU register set */
tag = l4_ipc_send(t1, u, tag, L4_IPC_NEVER);
if ((err = l4_ipc_error(tag, u)))
 printf("Error sending IPC: %x\n", err);

/* Idle around */
while (1)
 l4_sleep(10000);

return 0;
}

```

## 16.23 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTCB to store payload.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>,
 * Björn Döbel <doebel@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/util/thread.h>

#include <stdio.h>
#include <string.h>

static unsigned char stack2[8 << 10];
static l4_cap_idx_t thread1_cap, thread2_cap;

static void thread1(void)
{
 l4_msgregs_t *mr = l4_utcb_mr();
 l4_msgtag_t tag;
 int i, j;

 printf("Thread1 up (%p)\n", l4_utcb());

 for (i = 0; i < 10; i++)
 {
 for (j = 0; j < L4_UTCB_GENERIC_DATA_SIZE; j++)
 mr->mr[j] = 'A' + (i + j) % ('~' - 'A' + 1);
 tag = l4_msgtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
 if (l4_msgtag_has_error(l4_ipc_send(thread2_cap,
 l4_utcb(), tag, L4_IPC_NEVER)))
 printf("IPC-send error\n");
 }
}

L4UTIL_THREAD_STATIC_FUNC(thread2)
{
 l4_msgtag_t tag;
 l4_msgregs_t mr;
 unsigned i;

 printf("Thread2 up (%p)\n", l4_utcb());

 while (1)
 {
 if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap,
 l4_utcb(), L4_IPC_NEVER)))
 printf("IPC receive error\n");
 memcpy(&mr, l4_utcb_mr(), sizeof(mr));
 printf("Thread2 receive (%d): ", l4_msgtag_words(tag));
 for (i = 0; i < l4_msgtag_words(tag); i++)
 printf("%c", (char)mr.mr[i]);
 printf("\n");
 }

 __builtin_trap();
}

int main(void)
{
 l4_msgtag_t tag;

 thread1_cap = l4re_env()->main_thread;
 thread2_cap = l4re_util_cap_alloc();

 if (l4_is_invalid_cap(thread2_cap))
 return 1;

 tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
 if (l4_error(tag))

```

```

 return 1;

l4_thread_control_start();
l4_thread_control_pager(l4re_env()->rm);
l4_thread_control_exc_handler(l4re_env()->rm);
l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
 L4RE_THIS_TASK_CAP);
tag = l4_thread_control_commit(thread2_cap);
if (l4_error(tag))
 return 2;

tag = l4_thread_ex_regs(thread2_cap,
 (l4_umword_t)thread2,
 (l4_umword_t)(stack2 + sizeof(stack2)), 0);
if (l4_error(tag))
 return 3;

l4_sched_param_t sp = l4_sched_param(1, 0);
tag = l4_scheduler_run_thread(l4re_env()->scheduler, thread2_cap, &sp);
if (l4_error(tag))
 return 4;

thread1();

return 0;
}

```

## 16.24 examples/sys/ux-vhw/main.c

This example shows how to iterate the virtual hardware descriptors under Fiasco-UX.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/vhw.h>
#include <l4/util/util.h>
#include <l4/util/kip.h>
#include <l4/re/env.h>

#include <stdlib.h>
#include <stdio.h>

static void print_entry(struct l4_vhw_entry *e)
{
 printf("type: %d mem start: %08lx end: %08lx\n"
 "irq: %d pid %d\n",
 e->type, e->mem_start, e->mem_size,
 e->irq_no, e->provider_pid);
}

int main(void)
{
 l4_kernel_info_t *kip = l4re_kip();
 struct l4_vhw_descriptor *vhw;
 int i;

 if (!kip)
 {
 printf("KIP not available!\n");
 return 1;
 }

 if (!l4util_kip_kernel_is_ux(kip))
 {
 printf("This example is for Fiasco-UX only.\n");
 return 1;
 }

 vhw = l4_vhw_get(kip);

 printf("kip at %p, vhw at %p\n", kip, vhw);
 printf("magic: %08x, version: %08x, count: %02d\n",
 vhw->magic, vhw->version, vhw->count);
}

```

```

 for (i = 0; i < vhw->count; i++)
 print_entry(l4_vhw_get_entry(vhw, i));

 return 0;
}

```

## 16.25 hello/server/src/main.c

This is the famous "Hello World!" program.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
 * Lukas Grützmacher <lg2@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <unistd.h>

int
main(void)
{
 for (;;)
 {
 puts("Hello World!");
 sleep(1);
 }
}

```

## 16.26 tmpfs/lib/src/fs.cc

Example file system for [L4Re::Vfs](#).

```

/*
 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 * economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#include <l4/l4re_vfs/backend>
#include <l4/cxx/string>
#include <l4/cxx/avl_tree>

#include <sys/stat.h>
#include <sys/ioctl.h>
#include <dirent.h>

#include <cstdio>
#include <cstdlib>
#include <cstring>

namespace {

using namespace L4Re::Vfs;
using cxx::Ref_ptr;

class File_data
{
public:
 File_data() : _buf(0), _size(0) {}

 unsigned long put(unsigned long offset,
 unsigned long bufsize, void *srcbuf);
}

```

```

 unsigned long get(unsigned long offset,
 unsigned long bufsize, void *dstbuf);

 unsigned long size(unsigned long offset);
 unsigned long size() const { return _size; }

 ~File_data() throw() { free(_buf); }

private:
 void *_buf;
 unsigned long _size;
};

unsigned long
File_data::put(unsigned long offset, unsigned long bufsize, void *srcbuf)
{
 if (offset + bufsize > _size)
 size(offset + bufsize);

 if (!_buf)
 return 0;

 memcpy((char *)_buf + offset, srcbuf, bufsize);
 return bufsize;
}

unsigned long
File_data::get(unsigned long offset, unsigned long bufsize, void *dstbuf)
{
 unsigned long s = bufsize;

 if (offset > _size)
 return 0;

 if (offset + bufsize > _size)
 s = _size - offset;

 memcpy(dstbuf, (char *)_buf + offset, s);
 return s;
}

unsigned long
File_data::size(unsigned long offset)
{
 if (offset != _size)
 {
 _size = offset;
 _buf = realloc(_buf, _size);
 }

 if (!_buf)
 return 0;
 return -ENOSPC;
}

class Node : public cxx::Avl_tree_node
{
public:
 Node(const char *path, mode_t mode)
 : _ref_cnt(0), _path(strdup(path))
 {
 memset(&_info, 0, sizeof(_info));
 _info.st_mode = mode;
 }

 const char *path() const { return _path; }
 struct stat64 *info() { return &_info; }

 void add_ref() throw() { ++_ref_cnt; }
 int remove_ref() throw() { return --_ref_cnt; }

 bool is_dir() const { return S_ISDIR(_info.st_mode); }

 virtual ~Node() { free(_path); }

private:
 int _ref_cnt;
 char *_path;
 struct stat64 _info;
};

struct Node_get_key
{
 typedef cxx::String Key_type;
 static Key_type key_of(Node const *n)
 { return n->path(); }
};

```

```

};

struct Path_avl_tree_compare
{
 bool operator () (const char *l, const char *r) const
 { return strcmp(l, r) < 0; }
 bool operator () (const cxx::String l, const cxx::String r) const
 {
 int v = strncmp(l.start(), r.start(), cxx::min(l.len(), r.
 len()));
 return v < 0 || (v == 0 && l.len() < r.len());
 }
};

class Pers_file : public Node
{
public:
 Pers_file(const char *name, mode_t mode)
 : Node(name, (mode & 0777) | __S_IFREG) {}
 File_data const &data() const { return _data; }
 File_data &data() { return _data; }
private:
 File_data _data;
};

class Pers_dir : public Node
{
private:
 typedef cxx::Avl_tree<Node, Node_get_key, Path_avl_tree_compare>
 Tree;
 Tree _tree;

public:
 Pers_dir(const char *name, mode_t mode)
 : Node(name, (mode & 0777) | __S_IFDIR) {}
 Ref_ptr<Node> find_path(cxx::String);
 bool add_node(Ref_ptr<Node> const &);

 typedef Tree::Const_iterator Const_iterator;
 Const_iterator begin() const { return _tree.begin(); }
 Const_iterator end() const { return _tree.end(); }
};

Ref_ptr<Node> Pers_dir::find_path(cxx::String path)
{
 return cxx::ref_ptr(_tree.find_node(path));
}

bool Pers_dir::add_node(Ref_ptr<Node> const &n)
{
 bool e = _tree.insert(n.ptr()).second;
 if (e)
 n->add_ref();
 return e;
}

class Tmpfs_dir : public Be_file
{
public:
 explicit Tmpfs_dir(Ref_ptr<Pers_dir> const &d) throw()
 : _dir(d), _getdents_state(false) {}
 int get_entry(const char *, int, mode_t, Ref_ptr<File> *) throw();
 ssize_t getdents(char *, size_t) throw();
 int fstat64(struct stat64 *buf) const throw();
 int utime(const struct utimbuf *) throw();
 int fchmod(mode_t) throw();
 int mkdir(const char *, mode_t) throw();
 int unlink(const char *) throw();
 int rename(const char *, const char *) throw();
 int faccessat(const char *, int, int) throw();

private:
 int walk_path(cxx::String const &s,
 Ref_ptr<Node> *ret, cxx::String *remaining = 0);

 Ref_ptr<Pers_dir> _dir;
 bool _getdents_state;
 Pers_dir::Const_iterator _getdents_iter;
};

class Tmpfs_file : public Be_file_pos
{
public:
 explicit Tmpfs_file(Ref_ptr<Pers_file> const &f) throw()
 : Be_file_pos(), _file(f) {}

 off64_t size() const throw();

```

```

 int fstat64(struct stat64 *buf) const throw();
 int ftruncate64(off64_t p) throw();
 int ioctl(unsigned long, va_list) throw();
 int utime(const struct utimbuf *) throw();
 int fchmod(mode_t) throw();

private:
 ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) throw();
 ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) throw();
 Ref_ptr<Pers_file> _file;
};

ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p) throw()
{
 if (iovcnt < 0)
 return -EINVAL;

 ssize_t sum = 0;
 for (int i = 0; i < iovcnt; ++i)
 {
 sum += _file->data().get(p, v[i].iov_len, v[i].iov_base);
 p += v[i].iov_len;
 }
 return sum;
}

ssize_t Tmpfs_file::pwritev(const struct iovec *v, int iovcnt, off64_t p) throw()
{
 if (iovcnt < 0)
 return -EINVAL;

 ssize_t sum = 0;
 for (int i = 0; i < iovcnt; ++i)
 {
 sum += _file->data().put(p, v[i].iov_len, v[i].iov_base);
 p += v[i].iov_len;
 }
 return sum;
}

int Tmpfs_file::fstat64(struct stat64 *buf) const throw()
{
 _file->info()->st_size = _file->data().size();
 memcpy(buf, _file->info(), sizeof(*buf));
 return 0;
}

int Tmpfs_file::ftruncate64(off64_t p) throw()
{
 if (p < 0)
 return -EINVAL;

 if (_file->data().size(p) == 0)
 return 0;

 return -EIO; // most likely ENOSPC, but can't report that
}

off64_t Tmpfs_file::size() const throw()
{
 return _file->data().size();
}

int
Tmpfs_file::ioctl(unsigned long v, va_list args) throw()
{
 switch (v)
 {
 case FIONREAD: // return amount of data still available
 int *available = va_arg(args, int *);
 *available = _file->data().size() - pos();
 return 0;
 };
 return -EINVAL;
}

int
Tmpfs_file::utime(const struct utimbuf *times) throw()
{
 _file->info()->st_atime = times->actime;
 _file->info()->st_mtime = times->modtime;
 return 0;
}

int
Tmpfs_file::fchmod(mode_t m) throw()
{
 _file->info()->st_mode = m;
 return 0;
}

```



```

}

int
Tmpfs_dir::faccessat(const char *path, int mode, int) throw()
{
 Ref_ptr<Node> node;
 cxx::String name = path;

 int err = walk_path(name, &node, &name);
 if (err < 0)
 return err;

 if (mode == F_OK) // existence check
 return 0;

 struct stat64 *stats = node->info();

 if ((mode & R_OK) && !(stats->st_mode & S_IRUSR))
 return -EACCES;

 if ((mode & W_OK) && !(stats->st_mode & S_IWUSR))
 return -EACCES;

 if ((mode & X_OK) && !(stats->st_mode & S_IXUSR))
 return -EACCES;

 return 0;
}

int
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
 Ref_ptr<File> *file) throw()
{
 Ref_ptr<Node> path;
 if (!*name)
 {
 *file = cxx::ref_ptr(this);
 return 0;
 }

 cxx::String n = name;

 int e = walk_path(n, &path, &n);

 if (e == -ENOTDIR)
 return e;

 if (!(flags & O_CREAT) && e < 0)
 return e;

 if ((flags & O_CREAT) && e == -ENOENT)
 {
 Ref_ptr<Node> node(new Pers_file(n.start(), mode));
 // when ENOENT is return, path is always a directory
 bool e = cxx::ref_ptr_static_cast<Pers_dir>(path)->add_node(node);
 if (!e)
 return -ENOMEM;
 path = node;
 }

 if (path->is_dir())
 *file = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path)));
 else
 *file = cxx::ref_ptr(new Tmpfs_file(cxx::ref_ptr_static_cast<Pers_file>(path)));

 if (!*file)
 return -ENOMEM;

 return 0;
}

ssize_t
Tmpfs_dir::getdents(char *buf, size_t sz) throw()
{
 struct dirent64 *d = (struct dirent64 *)buf;
 ssize_t ret = 0;

 if (!_getdents_state)
 {
 _getdents_iter = _dir->begin();
 _getdents_state = true;
 }
 else if (_getdents_iter == _dir->end())
 {
 _getdents_state = false;
 return 0;
 }

```

```

 }

 for (; _getdents_iter != _dir->end(); ++_getdents_iter)
 {
 unsigned l = strlen(_getdents_iter->path()) + 1;
 if (l > sizeof(d->d_name))
 l = sizeof(d->d_name);

 unsigned n = offsetof (struct dirent64, d_name) + l;
 n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);

 if (n > sz)
 break;

 d->d_ino = 1;
 d->d_off = 0;
 memcpy(d->d_name, _getdents_iter->path(), l);
 d->d_reclen = n;
 d->d_type = DT_REG;
 ret += n;
 sz -= n;
 d = (struct dirent64 *) ((unsigned long)d + n);
 }

 return ret;
}

int
Tmpfs_dir::fstat64(struct stat64 *buf) const throw()
{
 memcpy(buf, _dir->info(), sizeof(*buf));
 return 0;
}

int
Tmpfs_dir::utime(const struct utimbuf *times) throw()
{
 _dir->info()->st_atime = times->actime;
 _dir->info()->st_mtime = times->modtime;
 return 0;
}

int
Tmpfs_dir::fchmod(mode_t m) throw()
{
 _dir->info()->st_mode = m;
 return 0;
}

int
Tmpfs_dir::walk_path(cxx::String const &_s,
 Ref_ptr<Node> *ret, cxx::String *remaining)
{
 Ref_ptr<Pers_dir> p = _dir;
 cxx::String s = _s;
 Ref_ptr<Node> n;

 while (1)
 {
 if (s.len() == 0)
 {
 *ret = p;
 return 0;
 }

 cxx::String::Index sep = s.find("/");

 if (sep - s.start() == 1 && *s.start() == '.')
 {
 s = s.substr(s.start() + 2);
 continue;
 }

 n = p->find_path(s.head(sep - s.start()));

 if (!n)
 {
 *ret = p;
 if (remaining)
 *remaining = s.head(sep - s.start());
 return -ENOENT;
 }

 if (sep == s.end())
 {
 *ret = n;

```

```

 return 0;
 }

 if (!n->is_dir())
 return -ENOTDIR;

 s = s.substr(sep + 1);

 p = cxx::ref_ptr_static_cast<Pers_dir>(n);
}

*ret = n;

return 0;
}

int
Tmpfs_dir::mkdir(const char *name, mode_t mode) throw()
{
 Ref_ptr<Node> node = _dir;
 cxx::String p = cxx::String(name);
 cxx::String path, last = p;
 cxx::String::Index s = p.rfind("/");

 // trim /'s at the end
 while (p.len() && s == p.end() - 1)
 {
 p.len(p.len() - 1);
 s = p.rfind("/");
 }

 //printf("MKDIR '%s' p=%p %p\n", name, p.start(), s);

 if (s != p.end())
 {
 path = p.head(s);
 last = p.substr(s + 1, p.end() - s);

 int e = walk_path(path, &node);
 if (e < 0)
 return e;
 }

 if (!node->is_dir())
 return -ENOTDIR;

 // due to path walking we can end up with an empty name
 if (p.len() == 0 || p == cxx::String("."))
 return 0;

 Ref_ptr<Pers_dir> dnode = cxx::ref_ptr_static_cast<Pers_dir>(node);

 Ref_ptr<Pers_dir> dir(new Pers_dir(last.start(), mode));
 return dnode->add_node(dir) ? 0 : -EEXIST;
}

int
Tmpfs_dir::unlink(const char *name) throw()
{
 cxx::Ref_ptr<Node> n;

 int e = walk_path(name, &n);
 if (e < 0)
 return -ENOENT;

 printf("Unimplemented (if file exists): %s(%s)\n", __func__, name);
 return -ENOMEM;
}

int
Tmpfs_dir::rename(const char *old, const char *newn) throw()
{
 printf("Unimplemented: %s(%s, %s)\n", __func__, old, newn);
 return -ENOMEM;
}

class Tmpfs_fs : public Be_file_system
{
public:
 Tmpfs_fs() : Be_file_system("tmpfs") {}
 int mount(char const *source, unsigned long mountflags,
 void const *data, cxx::Ref_ptr<File> *dir) throw()
 {
 (void)mountflags;
 (void)source;
 }

```

```
(void) data;
*dir = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr(new Pers_dir("root", 0777))));
if (!*dir)
 return -ENOMEM;
return 0;
}
};

static Tmpfs_fs _tmpfs L4RE_VFS_FILE_SYSTEM_ATTRIBUTE;
}
```

# Index

- \_\_lface
    - L4::Kobject\_2t, [1094](#)
    - L4::Kobject\_3t, [1098](#)
  - \_\_lface\_list
    - L4::Kobject\_2t, [1094](#)
    - L4::Kobject\_3t, [1098](#)
  - \_\_L4UTIL\_THREAD\_FUNC
    - l4/util/thread.h, [2328](#)
  - \_\_check\_protocols\_\_
    - L4::Kobject\_2t, [1095](#)
    - L4::Kobject\_3t, [1099](#)
  - \_bus
    - L4vbus::Device, [1539](#)
  - \_c
    - L4::Cap\_base, [896](#)
  - ~Auto\_ptr
    - cxx::Auto\_ptr, [697](#)
  - ~Be\_file\_system
    - L4Re::Vfs::Be\_file\_system, [1446](#)
  - ~H\_list\_item\_t
    - cxx::H\_list\_item\_t, [801](#)
- a
  - L4Re::Video::Pixel\_info, [1492](#)
- ARM Virtual Registers (UTCB), [91](#)
- Access\_width
  - L4Re::Mmio\_space, [1345](#)
- acked
  - L4virtio::Svr::Dev\_status, [1642](#), [1643](#)
- acked\_bfm\_t
  - L4virtio::Svr::Dev\_status, [1641](#)
- acquire
  - L4Re::Inhibitor, [1332](#)
- add
  - L4::lpc\_svr::Timeout\_queue, [1057](#)
  - L4::Thread::Modify\_senders, [1187](#)
  - L4vcpu::State, [1579](#)
  - L4virtio::Svr::Driver\_mem\_list\_t, [1654](#)
- add\_ku\_mem
  - L4::Task, [1167](#)
- add\_timeout
  - L4::lpc\_svr::Server\_iface, [1047](#)
  - L4::lpc\_svr::Timeout\_queue\_hooks, [1062](#)
- align\_to
  - L4::lpc::Msg, [662](#)
- all
  - L4::Kip::Mem\_desc, [1082](#), [1083](#)
- alloc
  - cxx::Base\_slab\_static, [721](#)
  - cxx::List\_alloc, [809](#)
  - cxx::Slab, [842](#)
  - cxx::Slab\_static, [846](#)
  - L4Re::Cap\_alloc, [1285](#), [1286](#)
  - L4Re::Mem\_alloc, [1341](#)
  - L4Re::Util::Counting\_cap\_alloc, [1386](#), [1387](#)
- alloc\_buffer\_demand
  - L4::lpc\_svr::Br\_manager\_no\_buffers, [1034](#)
  - L4::lpc\_svr::Server\_iface, [1047](#)
  - L4Re::Util::Br\_manager, [1377](#)
- alloc\_descriptor
  - L4virtio::Driver::Virtqueue, [1601](#)
- alloc\_dynamic\_entry
  - L4Re::Util::Names::Name\_space, [1419](#)
- alloc\_fd
  - L4Re::Vfs::Fs, [1459](#)
- alloc\_max
  - cxx::List\_alloc, [810](#)
- allocate
  - L4Re::Dataspace, [1294](#)
  - L4Re::Util::Dataspace\_svr, [1393](#)
- amd64 Virtual Registers (UTCB), [622](#)
- amd64/l4/sys/cache.h, [2066](#)
- amd64/l4/sys/consts.h, [2080](#), [2081](#)
- amd64/l4/sys/l4int.h, [2173](#), [2174](#)
- amd64/l4/sys/linkage.h, [1786](#)
- amd64/l4/sys/segment.h, [1761](#), [1765](#)
  - fiasco\_amd64\_segment\_info, [1763](#)
  - fiasco\_amd64\_set\_fs, [1764](#)
  - fiasco\_amd64\_set\_segment\_base, [1764](#)
  - L4\_task\_ldt\_x86\_consts, [1763](#)
- amd64/l4/sys/utcb.h, [2228](#), [2230](#)
- amd64/l4/util/apic.h, [1715](#), [1716](#)
- amd64/l4/util/bitops\_arch.h, [1792](#)
- amd64/l4/util/cpu.h, [1800](#), [1801](#)
- amd64/l4/util/idt.h, [1726](#), [1727](#)
- amd64/l4/util/irq.h, [1896](#), [1898](#)
  - l4util\_irq\_acknowledge, [1897](#)
- amd64/l4/util/l4\_macros.h, [1805](#)
- amd64/l4/util/mbi\_argv.h, [1808](#), [1809](#)
- amd64/l4/util/perform.h, [1729](#), [1730](#)
- amd64/l4/util/port\_io.h, [1771](#), [1772](#)
- amd64/l4/util/rdtsc.h, [1741](#), [1743](#)
- amd64/l4/util/spin.h, [1751](#)
- amd64/l4/util/stack\_impl.h, [1812](#), [1813](#)
  - l4util\_stack\_get\_sp, [1812](#)
- amd64/l4/util/util.h, [1753](#), [1755](#)
  - l4\_sleep, [1754](#)
  - l4util\_micros2l4to, [1754](#)
- amd64/l4f/l4/sys/segment.h, [1758](#), [1760](#)

- fiasco\_amd64\_set\_fs, 1759
  - fiasco\_amd64\_set\_segment\_base, 1760
- amd64/l4f/l4/util/port\_io.h, 1770, 1771
- amd64/l4f/l4/util/setjmp.h, 1779, 1781
  - l4\_thread\_longjmp, 1780
  - l4\_thread\_setjmp, 1781
- arm/l4/sys/atomic.h, 2261, 2262
- arm/l4/sys/cache.h, 2063, 2064
- arm/l4/sys/consts.h, 2079, 2080
- arm/l4/sys/l4int.h, 2173
- arm/l4/sys/linkage.h, 1785
- arm/l4/sys/mem\_op.h, 1787, 1789
- arm/l4/sys/thread.h, 2339
- arm/l4/sys/utcb.h, 2226, 2227
- arm/l4/sys/vm.h, 1790
- arm/l4/util/bitops\_arch.h, 1791, 1792
- arm/l4/util/cpu.h, 1799, 1800
- arm/l4/util/irq.h, 1895, 1896
- arm/l4/util/l4\_macros.h, 1804
- arm/l4/util/mbi\_argv.h, 1807, 1808
- arm/l4/util/stack\_impl.h, 1811, 1812
  - l4util\_stack\_get\_sp, 1811
- arm/l4f/l4/sys/ipc.h, 2151, 2152
- arm/l4f/l4/sys/syscall\_defs.h, 1814, 1815
- assign\_dma\_domain
  - L4vbus::Vbus, 1572, 1573
- associate
  - L4Re::Dma\_space, 1306
- Atomic Instructions, 92
  - l4util\_add8, 94
  - l4util\_add8\_res, 94
  - l4util\_atomic\_add, 94
  - l4util\_atomic\_inc, 95
  - l4util\_cmpxchg, 95
  - l4util\_cmpxchg16, 96
  - l4util\_cmpxchg32, 96
  - l4util\_cmpxchg64, 98
  - l4util\_cmpxchg8, 98
  - l4util\_inc8, 99
  - l4util\_inc8\_res, 99
  - l4util\_xchg, 99
  - l4util\_xchg16, 100
  - l4util\_xchg32, 100
  - l4util\_xchg8, 101
- attach
  - L4::Irq, 1066
  - L4Re::Rm, 1362, 1363
  - L4Re::Util::Event\_buffer\_t, 1407
- Attach\_flags
  - L4Re::Rm, 1361
- Attr
  - L4::Thread::Attr, 1184
- Attribute
  - L4Re::Dma\_space, 1305
- Attributes
  - L4Re::Dma\_space, 1304
- Auto\_ptr
  - cxx::Auto\_ptr, 697
- auto\_refresh
  - L4Re::Video::Goos::Info, 1489
- Auxiliary data, 102
- avail
  - cxx::List\_alloc, 811
- avail\_align
  - L4virtio::Virtqueue, 1684
- avail\_size
  - L4virtio::Virtqueue, 1684
- Avl\_map
  - cxx::Avl\_map, 702
- ax
  - l4\_vcpu\_regs\_t, 1269
- b
  - L4Re::Video::Pixel\_info, 1493
- backtrace.h
  - l4util\_backtrace, 2264
- Bad\_descriptor
  - L4virtio::Svr::Bad\_descriptor, 1609
- Base API, 103
- Base\_avl\_set
  - cxx::Bits::Base\_avl\_set, 760
- Basic Macros, 106
  - L4\_ALWAYS\_INLINE, 107
  - L4\_HIDDEN, 107
  - L4\_NOTHROW, 107
- Be\_file\_system
  - L4Re::Vfs::Be\_file\_system, 1446
- begin
  - cxx::Bits::Base\_avl\_set, 761
  - cxx::Bits::Bst, 775, 776
- bind
  - L4::lcu, 933
  - L4::lommu, 949
  - L4::Thread::Attr, 1184
- bind\_thread
  - L4::Rcv\_endpoint, 1130
- bit
  - cxx::Bitmap\_base, 745
- Bit Manipulation, 109
  - l4util\_bsf, 109
  - l4util\_bsr, 110
  - l4util\_btc, 111
  - l4util\_btr, 111
  - l4util\_bts, 112
  - l4util\_clear\_bit, 113
  - l4util\_complement\_bit, 113
  - l4util\_find\_first\_set\_bit, 114
  - l4util\_find\_first\_zero\_bit, 114
  - l4util\_next\_power2, 114
  - l4util\_set\_bit, 115
  - l4util\_test\_bit, 115
- bit\_index
  - cxx::Bitmap\_base, 746
- Bitmap
  - cxx::Bitmap, 740
- bits\_per\_pixel
  - L4Re::Video::Pixel\_info, 1493

- Bits\_type
  - cxx::Bitfield, [724](#)
- blk\_size
  - l4virtio\_block\_config\_t, [1708](#)
- Block\_dev
  - L4virtio::Svr::Block\_dev, [1614](#)
- bp
  - l4\_vcpu\_regs\_t, [1270](#)
- buf
  - L4Re::Util::Event\_buffer\_t, [1408](#)
- buf\_cp\_in
  - L4::lpc, [653](#)
- buf\_cp\_out
  - L4::lpc, [654](#)
- buf\_in
  - L4::lpc, [654](#)
- buffer
  - L4Re::Util::Event\_t, [1412](#)
- Buffer Registers (BRs), [117](#)
  - l4\_buffer\_desc\_consts\_t, [117](#)
- bus\_cap
  - L4vbus::Device, [1533](#)
- bx
  - l4\_vcpu\_regs\_t, [1270](#)
- bytes\_per\_pixel
  - L4Re::Video::Pixel\_info, [1494](#)
- c
  - L4::Kobject\_2t, [1095](#)
  - L4::Kobject\_3t, [1099](#)
- C++ Exceptions, [119](#)
- C++ IPC Interface Definition., [120](#)
- CPU related functions, [121](#)
  - l4util\_cpu\_capabilities, [121](#)
  - l4util\_cpu\_capabilities\_nocheck, [122](#)
  - l4util\_cpu\_has\_cpuid, [123](#)
- Cache Consistency, [124](#)
  - l4\_cache\_clean\_data, [124](#)
  - l4\_cache\_coherent, [125](#)
  - l4\_cache\_dma\_coherent, [125](#)
  - l4\_cache\_flush\_data, [126](#)
  - l4\_cache\_inv\_data, [126](#)
- call
  - L4::Arm\_smccc, [873](#)
  - L4::lpc::loststream, [973](#)
- Cap
  - L4::Cap, [883](#), [884](#)
  - L4::lpc::Cap, [964](#)
- cap
  - L4::Cap\_base, [889](#)
  - L4::Invalid\_capability, [943](#)
  - L4::Kobject, [1089](#)
- cap\_alloc
  - L4Re Capability API, [376](#)
- Cap\_base
  - L4::Cap\_base, [888](#), [889](#)
- cap\_cast
  - L4, [644](#), [645](#)
- cap\_dynamic\_cast
  - L4, [646](#)
- cap\_equal
  - L4::Task, [1168](#)
- cap\_has\_child
  - L4::Task, [1168](#)
- cap\_received
  - L4::lpc::Gen\_fpage, [967](#)
- cap\_reinterpret\_cast
  - L4, [647](#), [648](#)
- Cap\_type
  - L4::Cap\_base, [888](#)
- cap\_valid
  - L4::Task, [1169](#)
- Capabilities, [128](#)
  - l4\_cap\_consts\_t, [129](#)
  - l4\_capability\_equal, [130](#)
  - l4\_default\_caps\_t, [129](#)
  - l4\_is\_invalid\_cap, [130](#)
  - l4\_is\_valid\_cap, [131](#)
- capability
  - L4\_DISABLE\_COPY, [2070](#)
- Capability allocator, [132](#)
  - l4re\_util\_cap\_last, [132](#)
- cast
  - L4vcpu::Vcpu, [1584](#)
- cfg\_read
  - L4vbus::Pci\_dev, [1560](#)
  - L4vbus::Pci\_host\_bridge, [1565](#)
- cfg\_write
  - L4vbus::Pci\_dev, [1560](#)
  - L4vbus::Pci\_host\_bridge, [1566](#)
- chain
  - L4::lrc\_mux, [1078](#)
- change\_queue\_config
  - L4virtio::Svr::Dev\_config, [1628](#)
- chars
  - cxx::Bitmap\_base, [747](#)
- check\_size
  - L4::lpc::Msg, [663](#), [664](#)
- chkcap
  - L4Re, [674](#)
- chkipc
  - L4Re, [674](#)
- chksys
  - L4Re, [676–678](#)
- Chunks, [133](#)
  - l4shmc\_add\_chunk, [133](#)
  - l4shmc\_chunk\_capacity, [134](#)
  - l4shmc\_chunk\_ptr, [134](#)
  - l4shmc\_chunk\_signal, [135](#)
  - l4shmc\_get\_chunk, [135](#)
  - l4shmc\_get\_chunk\_to, [136](#)
  - l4shmc\_iterate\_chunk, [136](#)
- Class
  - L4::Kobject\_2t, [1094](#)
  - L4::Kobject\_3t, [1098](#)
- clear
  - cxx::Bits::Basic\_list, [770](#)

- L4::Types::Flags, [1224](#)
  - L4Re::Dataspace, [1294](#)
  - L4Re::Util::Dataspace\_svr, [1394](#)
  - L4vcpu::State, [1579](#)
- clear\_bit
  - cxx::Bitmap\_base, [747](#)
- cnt\_jobmap\_tlb\_flush
  - l4\_tracebuffer\_status\_t, [1264](#)
- Color\_component
  - L4Re::Video::Color\_component, [1479](#)
- Com\_error
  - L4::Com\_error, [899](#)
- Comfortable Command Line Parsing, [137](#)
  - parse\_cmdline, [137](#)
- config\_get
  - L4vbus::Gpio\_pin, [1550](#)
- config\_pad
  - L4vbus::Gpio\_module, [1543](#)
  - L4vbus::Gpio\_pin, [1551](#)
- config\_pull
  - L4vbus::Gpio\_pin, [1551](#)
- config\_queue
  - L4virtio::Device, [1595](#)
- Console API, [140](#)
- consumed
  - L4virtio::Svr::Virtqueue, [1675](#)
- Consumer, [141](#), [145](#)
  - l4shmc\_chunk\_consumed, [141](#)
  - l4shmc\_chunk\_size, [142](#)
  - l4shmc\_enable\_chunk, [142](#)
  - l4shmc\_enable\_signal, [145](#)
  - l4shmc\_is\_chunk\_ready, [143](#)
  - l4shmc\_wait\_any, [146](#)
  - l4shmc\_wait\_any\_to, [146](#)
  - l4shmc\_wait\_any\_try, [146](#)
  - l4shmc\_wait\_chunk, [143](#)
  - l4shmc\_wait\_chunk\_to, [143](#)
  - l4shmc\_wait\_chunk\_try, [144](#)
  - l4shmc\_wait\_signal, [147](#)
  - l4shmc\_wait\_signal\_to, [147](#)
  - l4shmc\_wait\_signal\_try, [148](#)
- contains
  - L4virtio::Svr::Driver\_mem\_region\_t, [1662](#)
- contrib/libio-io/l4/io/io.h, [1815](#), [1818](#)
- control
  - L4::Thread, [1176](#)
- copy
  - L4::Cap, [885](#)
  - L4::Cap\_base, [890](#)
  - L4Re::Util::Dataspace\_svr, [1394](#)
- copy\_in
  - L4Re::Dataspace, [1295](#)
- copy\_receive\_cap
  - L4Re::Util::Names::Name\_space, [1419](#)
- copy\_to
  - L4virtio::Svr::Data\_buffer, [1623](#)
- count
  - L4::Kip::Mem\_desc, [1083](#), [1084](#)
  - l4\_vhw\_descriptor, [1276](#)
- Counting\_cap\_alloc
  - L4Re::Util::Counting\_cap\_alloc, [1386](#)
- cpu\_disable
  - L4::Platform\_control, [1120](#)
- cpu\_enable
  - L4::Platform\_control, [1121](#)
- create
  - L4::Factory, [916](#), [917](#)
- create\_buffer
  - L4Re::Video::Goos, [1485](#)
- create\_factory
  - L4::Factory, [918](#)
- create\_gate
  - L4::Factory, [919](#)
- create\_irq
  - L4::Factory, [920](#)
- create\_task
  - L4::Factory, [921](#)
- create\_thread
  - L4::Factory, [922](#)
- create\_view
  - L4Re::Video::Goos, [1485](#)
- create\_vm
  - L4::Factory, [923](#)
- current\_flags
  - L4virtio::Svr::Request\_processor, [1667](#)
- cx
  - l4\_vcpu\_regs\_t, [1270](#)
- cxx, [637](#)
  - H\_list\_item, [639](#)
- cxx::Auto\_ptr
  - ~Auto\_ptr, [697](#)
  - Auto\_ptr, [697](#)
  - get, [697](#)
  - operator Priv\_type \*, [698](#)
  - operator\*, [698](#)
  - operator->, [698](#)
  - operator=, [698](#)
  - Ref\_type, [696](#)
  - release, [699](#)
- cxx::Auto\_ptr< T >, [695](#)
- cxx::Avl\_map
  - Avl\_map, [702](#)
  - insert, [703](#)
  - operator[], [703](#), [704](#)
- cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >, [700](#)
- cxx::Avl\_set< ITEM\_TYPE, COMPARE, ALLOC >, [704](#)
- cxx::Avl\_tree
  - insert, [711](#)
  - remove, [711](#)
- cxx::Avl\_tree< Node, Get\_key, Compare >, [707](#)
- cxx::Avl\_tree\_node, [712](#)
- cxx::Base\_slab
  - free\_objects, [717](#)
  - total\_objects, [717](#)



- cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc  
>, [715](#)
- cxx::Base\_slab\_static
  - alloc, [721](#)
  - free, [721](#)
  - free\_objects, [722](#)
  - total\_objects, [722](#)
- cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free,  
Alloc >, [718](#)
- cxx::Bitfield
  - Bits\_type, [724](#)
  - get, [726](#)
  - get\_unshifted, [727](#)
  - Masks, [726](#)
  - Ref, [724](#)
  - Ref\_unshifted, [725](#)
  - set, [727](#)
  - set\_dirty, [728](#)
  - set\_unshifted, [729](#)
  - set\_unshifted\_dirty, [730](#)
  - Shift\_type, [725](#)
  - Val, [725](#)
  - val, [731](#)
  - val\_dirty, [731](#)
  - Val\_unshifted, [725](#)
  - val\_unshifted, [732](#)
- cxx::Bitfield< T, LSB, MSB >, [723](#)
- cxx::Bitfield< T, LSB, MSB >::Value< TT >, [733](#)
- cxx::Bitfield< T, LSB, MSB >::Value\_base< TT >, [734](#)
- cxx::Bitfield< T, LSB, MSB >::Value\_unshifted< TT >,  
[736](#)
- cxx::Bitmap
  - Bitmap, [740](#)
  - scan\_zero, [740](#)
- cxx::Bitmap< BITS >, [737](#)
- cxx::Bitmap\_base, [741](#)
  - bit, [745](#)
  - bit\_index, [746](#)
  - chars, [747](#)
  - clear\_bit, [747](#)
  - operator[], [748](#)
  - scan\_zero, [749](#)
  - set\_bit, [750](#)
  - word\_index, [751](#)
  - words, [751](#)
- cxx::Bitmap\_base::Bit, [752](#)
- cxx::Bitmap\_base::Char< BITS >, [753](#)
- cxx::Bitmap\_base::Word< BITS >, [753](#)
- cxx::Bits, [639](#)
- cxx::Bits::Avl\_map\_get\_key< KEY\_TYPE >, [755](#)
- cxx::Bits::Avl\_set\_get\_key< KEY\_TYPE >, [755](#)
- cxx::Bits::Base\_avl\_set
  - Base\_avl\_set, [760](#)
  - begin, [761](#)
  - end, [762](#)
  - erase, [763](#)
  - find\_node, [763](#)
  - insert, [764](#)
  - lower\_bound\_node, [764](#)
  - rbegin, [764](#), [765](#)
  - remove, [765](#)
  - rend, [766](#)
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, A↵,  
LLOC, GET\_KEY >, [756](#)
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, A↵,  
LLOC, GET\_KEY >::Node, [767](#)
- cxx::Bits::Base\_avl\_set::Node
  - operator\*, [768](#)
  - operator->, [768](#)
  - valid, [768](#)
- cxx::Bits::Basic\_list
  - clear, [770](#)
  - iter, [770](#)
- cxx::Bits::Basic\_list< POLICY >, [769](#)
- cxx::Bits::Bst
  - begin, [775](#), [776](#)
  - dir, [776](#), [777](#)
  - end, [778](#)
  - find, [779](#)
  - find\_node, [780](#)
  - lower\_bound\_node, [780](#)
  - rbegin, [781](#), [782](#)
  - remove\_all, [782](#)
  - remove\_tree, [783](#)
  - rend, [783](#), [784](#)
- cxx::Bits::Bst< Node, Get\_key, Compare >, [771](#)
- cxx::Bits::Bst\_node, [785](#)
- cxx::Bits::Direction, [787](#)
  - Direction\_e, [788](#)
  - operator!, [788](#)
- cxx::Bits::Smart\_ptr\_list
  - pop\_front, [790](#)
- cxx::Bits::Smart\_ptr\_list< ITEM >, [789](#)
- cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T >, [791](#)
- cxx::H\_list
  - erase, [795](#)
  - insert, [796](#)
  - insert\_after, [796](#)
  - insert\_before, [797](#)
  - iter, [797](#)
  - pop\_front, [797](#)
  - remove, [798](#)
  - replace, [798](#)
- cxx::H\_list< T, POLICY >, [792](#)
- cxx::H\_list\_item\_t
  - ~H\_list\_item\_t, [801](#)
  - H\_list\_item\_t, [800](#)
- cxx::H\_list\_item\_t< ELEM\_TYPE >, [799](#)
- cxx::H\_list\_t< T >, [801](#)
- cxx::List
  - items, [805](#)
  - operator[], [805](#)
  - push\_back, [805](#)
  - push\_front, [806](#)
  - remove, [806](#)
  - size, [806](#)

cxx::List< D, Alloc >, 804  
 cxx::List< D, Alloc >::lter, 807  
 cxx::List\_alloc, 808  
     alloc, 809  
     alloc\_max, 810  
     avail, 811  
     free, 811  
     List\_alloc, 808  
 cxx::List\_item, 812  
     get\_next\_item, 814  
     get\_prev\_item, 814  
     insert\_next\_item, 814  
     insert\_prev\_item, 815  
     push\_back, 815  
     push\_front, 816  
     remove, 816  
     remove\_me, 817  
 cxx::List\_item::lter, 818  
     remove\_me, 820  
 cxx::List\_item::T\_iter< T, Poly >, 821  
 cxx::Lt\_functor< Obj >, 824  
 cxx::New\_allocator< \_Type >, 824  
 cxx::Nothrow, 825  
 cxx::Pair  
     Pair, 827  
 cxx::Pair< First, Second >, 826  
 cxx::Pair\_first\_compare  
     operator(), 828  
     Pair\_first\_compare, 828  
 cxx::Pair\_first\_compare< Cmp, Typ >, 827  
 cxx::Ref\_obj\_list\_item< T >, 829  
 cxx::Ref\_ptr  
     get, 834  
     ptr, 835  
     Ref\_ptr, 833, 834  
     release, 835  
 cxx::Ref\_ptr< T, CNT >, 830  
 cxx::S\_list  
     pop\_front, 839  
 cxx::S\_list< T, POLICY >, 836  
 cxx::Slab  
     alloc, 842  
     free, 842  
 cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, 839  
 cxx::Slab\_static  
     alloc, 846  
 cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >,  
     843  
 cxx::String, 848  
     find, 851  
     from\_dec, 852  
     from\_hex, 853  
     starts\_with, 854  
     String, 851  
 cxx::Weak\_ref< T >, 855  
 cxx::Weak\_ref\_base, 857  
 cxx::static\_vector< T, IDX >, 846  
  
 Elf32\_Dyn, 860  
 Elf64\_Dyn, 867  
 DF\_1\_CONFALT  
     ELF binary format, 178  
 DF\_1\_DIRECT  
     ELF binary format, 178  
 DF\_1\_DISPRELDNE  
     ELF binary format, 179  
 DF\_1\_DISPRELPND  
     ELF binary format, 179  
 DF\_1\_ENDFILTEE  
     ELF binary format, 179  
 DF\_1\_GLOBAL  
     ELF binary format, 179  
 DF\_1\_GROUP  
     ELF binary format, 179  
 DF\_1\_INTERPOSE  
     ELF binary format, 180  
 DF\_1\_LOADFLTR  
     ELF binary format, 180  
 DF\_1\_NODEFLIB  
     ELF binary format, 180  
 DF\_1\_NODELETE  
     ELF binary format, 180  
 DF\_1\_NODUMP  
     ELF binary format, 180  
 DF\_1\_NOOPEN  
     ELF binary format, 181  
 DF\_1\_NOW  
     ELF binary format, 181  
 DF\_1\_ORIGIN  
     ELF binary format, 181  
 DF\_P1\_GROUPPERM  
     ELF binary format, 181  
 DF\_P1\_LAZYLOAD  
     ELF binary format, 181  
 DMA API for L4Re, 149  
 DMA Space Interface, 150  
     l4re\_dma\_space\_associate, 151  
     l4re\_dma\_space\_disassociate, 151  
     l4re\_dma\_space\_map, 152  
     l4re\_dma\_space\_t, 150  
     l4re\_dma\_space\_unmap, 153  
 DT\_HIPROC  
     ELF binary format, 182  
 DT\_LOPROC  
     ELF binary format, 182  
 DT\_NULL  
     ELF binary format, 182  
 data  
     L4::lpc::Varg, 1021  
 Data\_buffer  
     L4virtio::Svr::Data\_buffer, 1623  
 data\_size  
     L4virtio::Svr::Block\_request, 1620  
 data\_space  
     L4Re::Vfs::Be\_file, 1443  
     L4Re::Vfs::Regular\_file, 1472

- Dataspace interface, [154](#)
  - [l4re\\_ds\\_allocate](#), [155](#)
  - [l4re\\_ds\\_clear](#), [155](#)
  - [l4re\\_ds\\_copy\\_in](#), [156](#)
  - [l4re\\_ds\\_flags](#), [156](#)
  - [l4re\\_ds\\_info](#), [156](#)
  - [l4re\\_ds\\_map\\_flags](#), [155](#)
  - [l4re\\_ds\\_phys](#), [157](#)
  - [l4re\\_ds\\_size](#), [157](#)
- debug
  - [L4Re::Debug\\_obj](#), [1303](#)
- Debug interface, [159](#)
  - [l4re\\_debug\\_obj\\_debug](#), [159](#)
- debugger.h
  - [l4\\_debugger\\_get\\_object\\_name](#), [2119](#)
  - [l4\\_debugger\\_query\\_log\\_name](#), [2119](#)
  - [l4\\_debugger\\_query\\_log\\_typeid](#), [2120](#)
  - [l4\\_debugger\\_switch\\_log](#), [2120](#)
- Debugging API, [160](#)
- dec\_refcnt
  - [L4::Kobject](#), [1091](#)
- delete\_buffer
  - [L4Re::Video::Goos](#), [1485](#)
- delete\_obj
  - [L4::Task](#), [1169](#)
- delete\_view
  - [L4Re::Video::Goos](#), [1486](#)
- Demand
  - [L4::Kobject\\_typeid](#), [1103](#)
  - [L4::Type\\_info::Demand](#), [1194](#)
- demand
  - [L4::Kobject\\_typeid](#), [1103](#)
- desc
  - [L4virtio::Driver::Virtqueue](#), [1601](#)
  - [L4virtio::Svr::Virtqueue](#), [1676](#)
  - [L4virtio::Svr::Virtqueue::Head\\_desc](#), [1679](#)
- desc\_align
  - [L4virtio::Virtqueue](#), [1685](#)
- desc\_avail
  - [L4virtio::Svr::Virtqueue](#), [1676](#)
- desc\_size
  - [L4virtio::Virtqueue](#), [1685](#)
- descs
  - [l4\\_vhw\\_descriptor](#), [1276](#)
- detach
  - [L4::Irq](#), [1067](#)
  - [L4Re::Rm](#), [1365](#), [1366](#)
  - [L4Re::Util::Event\\_buffer\\_t](#), [1408](#)
- Detach\_flags
  - [L4Re::Rm](#), [1361](#)
- Detach\_result
  - [L4Re::Rm](#), [1361](#)
- Dev\_config
  - [L4virtio::Svr::Dev\\_config](#), [1627](#)
- dev\_handle
  - [L4vbus::Device](#), [1534](#)
- device
  - [L4vbus::Device](#), [1534](#)
- device\_by\_hid
  - [L4vbus::Device](#), [1536](#)
- device\_config
  - [L4virtio::Device](#), [1596](#)
- device\_error
  - [L4virtio::Svr::Device\\_t](#), [1649](#)
- device\_notification\_irq
  - [L4virtio::Device](#), [1596](#)
- device\_notify\_irq
  - [L4virtio::Svr::Device\\_t](#), [1649](#)
- di
  - [l4\\_vcpu\\_regs\\_t](#), [1270](#)
- dir
  - [cxx::Bits::Bst](#), [776](#), [777](#)
- Direction
  - [L4Re::Dma\\_space](#), [1305](#)
- Direction\_e
  - [cxx::Bits::Direction](#), [788](#)
- disable
  - [L4virtio::Virtqueue](#), [1686](#)
- disable\_notify
  - [L4virtio::Svr::Virtqueue](#), [1677](#)
- disassociate
  - [L4Re::Dma\\_space](#), [1306](#)
- dispatch
  - [L4::Basic\\_registry](#), [877](#)
  - [L4::Server\\_object](#), [1153](#)
- dispatch\_meta\_request
  - [L4::Server\\_object\\_t](#), [1156](#)
- dma\_space.h
  - [l4re\\_dma\\_space\\_direction](#), [1913](#)
  - [l4re\\_dma\\_space\\_dma\\_addr\\_t](#), [1913](#)
  - [l4re\\_dma\\_space\\_space\\_attribs](#), [1914](#)
- done
  - [L4virtio::Svr::Data\\_buffer](#), [1624](#)
- down
  - [L4::Semaphore](#), [1147](#)
- drive\_cylinders
  - [l4util\\_mb\\_drive\\_t](#), [1523](#)
- drive\_mode
  - [l4util\\_mb\\_drive\\_t](#), [1523](#)
- drive\_number
  - [l4util\\_mb\\_drive\\_t](#), [1524](#)
- driver
  - [L4virtio::Svr::Dev\\_status](#), [1643](#)
- driver\_bfm\_t
  - [L4virtio::Svr::Dev\\_status](#), [1641](#)
- Driver\_mem\_region\_t
  - [L4virtio::Svr::Driver\\_mem\\_region\\_t](#), [1662](#)
- driver\_ok
  - [L4virtio::Svr::Dev\\_status](#), [1644](#)
- driver\_ok\_bfm\_t
  - [L4virtio::Svr::Dev\\_status](#), [1642](#)
- drv\_base
  - [L4virtio::Svr::Driver\\_mem\\_region\\_t](#), [1663](#)
- ds
  - [L4virtio::Svr::Dev\\_config](#), [1629](#)
  - [L4virtio::Svr::Driver\\_mem\\_region\\_t](#), [1663](#)

- ds\_offset
  - L4virtio::Svr::Driver\_mem\_region\_t, 1663
- dump
  - L4Re::Video::Color\_component, 1479
  - L4Re::Video::Pixel\_info, 1494
  - L4virtio::Virtqueue, 1686
- dx
  - l4\_vcpu\_regs\_t, 1271
- e\_phnum
  - Elf32\_Ehdr, 862
  - Elf64\_Ehdr, 868
- e\_shnum
  - Elf32\_Ehdr, 862
  - Elf64\_Ehdr, 869
- EDID parsing functionality, 161
  - libedid\_check\_header, 162
  - libedid\_checksum, 162
  - Libedid\_consts, 161
  - libedid\_dump, 162
  - libedid\_dump\_standard\_timings, 163
  - libedid\_num\_ext\_blocks, 163
  - libedid\_pnp\_id, 163
  - libedid\_prefered\_resolution, 164
  - libedid\_revision, 164
  - libedid\_version, 164
- EI\_CLASS
  - ELF binary format, 182
- EI\_DATA
  - ELF binary format, 183
- EI\_OSABI
  - ELF binary format, 183
- EI\_PAD
  - ELF binary format, 184
- EI\_VERSION
  - ELF binary format, 184
- ELF binary format, 166
  - DF\_1\_CONFALT, 178
  - DF\_1\_DIRECT, 178
  - DF\_1\_DISPRELDNE, 179
  - DF\_1\_DISPRELPND, 179
  - DF\_1\_ENDFILTEE, 179
  - DF\_1\_GLOBAL, 179
  - DF\_1\_GROUP, 179
  - DF\_1\_INTERPOSE, 180
  - DF\_1\_LOADFLTR, 180
  - DF\_1\_NODEFLIB, 180
  - DF\_1\_NODELETE, 180
  - DF\_1\_NODUMP, 180
  - DF\_1\_NOOPEN, 181
  - DF\_1\_NOW, 181
  - DF\_1\_ORIGIN, 181
  - DF\_P1\_GROUPPER, 181
  - DF\_P1\_LAZYLOAD, 181
  - DT\_HIPROC, 182
  - DT\_LOPROC, 182
  - DT\_NULL, 182
  - EI\_CLASS, 182
  - EI\_DATA, 183
  - EI\_OSABI, 183
  - EI\_PAD, 184
  - EI\_VERSION, 184
  - ELFCLASSNONE, 185
  - ELFDATA2LSB, 185
  - ELFDATA2MSB, 186
  - ELFDATANONE, 186
  - ELFOSABI\_AIX, 187
  - ELFOSABI\_FREEBSD, 187
  - ELFOSABI\_HPUX, 187
  - ELFOSABI\_IRIX, 187
  - ELFOSABI\_LINUX, 188
  - ELFOSABI\_MODESTO, 188
  - ELFOSABI\_NETBSD, 188
  - ELFOSABI\_OPENBSD, 188
  - ELFOSABI\_SOLARIS, 188
  - ELFOSABI\_SYSV, 189
  - ELFOSABI\_TRU64, 189
  - EM\_ARC, 189
  - NT\_VERSION, 189
  - PT\_GNU\_EH\_FRAME, 190
  - PT\_GNU\_RELRO, 190
  - PT\_GNU\_STACK, 190
  - PT\_HIOS, 190
  - PT\_HIPROC, 190
  - PT\_L4\_AUX, 191
  - PT\_L4\_KIP, 191
  - PT\_L4\_STACK, 191
  - PT\_LOOS, 191
  - PT\_LOPROC, 191
  - SHF\_GROUP, 192
  - SHF\_MASKOS, 192
  - SHF\_TLS, 192
  - SHT\_NUM, 192

- ELFOSABI\_SOLARIS
  - ELF binary format, [188](#)
- ELFOSABI\_SYSV
  - ELF binary format, [189](#)
- ELFOSABI\_TRU64
  - ELF binary format, [189](#)
- EM\_ARC
  - ELF binary format, [189](#)
- Elf32\_Dyn, [860](#)
  - d\_val, [860](#)
- Elf32\_Ehdr, [861](#)
  - e\_phnum, [862](#)
  - e\_shnum, [862](#)
- Elf32\_Phdr, [863](#)
- Elf32\_Shdr, [864](#)
- Elf32\_Sym, [865](#)
- Elf64\_Dyn, [866](#)
  - d\_val, [867](#)
- Elf64\_Ehdr, [867](#)
  - e\_phnum, [868](#)
  - e\_shnum, [869](#)
- Elf64\_Phdr, [869](#)
- Elf64\_Shdr, [870](#)
- Elf64\_Sym, [871](#)
- empty
  - L4virtio::Svr::Driver\_mem\_region\_t, [1663](#)
- enable\_notify
  - L4virtio::Svr::Virtqueue, [1677](#)
- end
  - cxx::Bits::Base\_avl\_set, [762](#)
  - cxx::Bits::Bst, [778](#)
  - L4::Kip::Mem\_desc, [1084](#)
- enqueue\_descriptor
  - L4virtio::Driver::Virtqueue, [1602](#)
- entry\_ip
  - L4vcpu::Vcpu, [1584](#)
- entry\_sp
  - L4vcpu::Vcpu, [1585](#)
- env
  - L4Re::Env, [1311](#)
- env.h
  - l4re\_env\_t, [1967](#)
- erase
  - cxx::Bits::Base\_avl\_set, [763](#)
  - cxx::H\_list, [795](#)
- err\_no
  - L4::Runtime\_error, [1138](#)
- Error
  - L4virtio::Svr::Bad\_descriptor, [1609](#)
- Error codes, [200](#)
  - l4\_error\_code\_t, [200](#)
- Error Handling, [193](#)
  - l4\_error, [194](#)
  - l4\_ipc\_error, [195](#)
  - l4\_ipc\_error\_code, [197](#)
  - l4\_ipc\_is\_rcv\_error, [198](#)
  - l4\_ipc\_is\_snd\_error, [198](#)
  - l4\_ipc\_tcr\_error\_t, [194](#)
- Event API, [202](#)
- Event interface, [203](#)
  - l4re\_event\_get\_axis\_info, [203](#)
  - l4re\_event\_get\_buffer, [204](#)
  - l4re\_event\_get\_num\_streams, [204](#)
  - l4re\_event\_get\_stream\_info, [205](#)
  - l4re\_event\_get\_stream\_info\_for\_id, [205](#)
- Event\_buffer\_t
  - L4Re::Event\_buffer\_t, [1326](#)
- ex\_regs
  - L4::Thread, [1176](#), [1177](#)
- exc\_handler
  - L4::Thread::Attr, [1185](#)
- Exception registers, [207](#)
  - l4\_utcb\_exc, [207](#)
  - l4\_utcb\_exc\_is\_ex\_regs\_exception, [207](#)
  - l4\_utcb\_exc\_is\_pf, [208](#)
  - l4\_utcb\_exc\_pc, [208](#)
  - l4\_utcb\_exc\_pc\_set, [209](#)
- execute
  - L4Re::Ned::Cmd\_control, [1354](#)
- expired
  - L4::lpc\_svr::Timeout, [1055](#)
- ext\_alloc
  - L4vcpu::Vcpu, [1585](#)
- Extended vCPU support, [210](#)
  - l4vcpu\_ext\_alloc, [210](#)
- extra\_str
  - L4::Runtime\_error, [1138](#)
- faccessat
  - L4Re::Vfs::Directory, [1449](#)
- Factory, [212](#)
  - l4\_factory\_create\_factory, [213](#)
  - l4\_factory\_create\_factory\_u, [214](#)
  - l4\_factory\_create\_gate, [215](#)
  - l4\_factory\_create\_gate\_u, [216](#)
  - l4\_factory\_create\_irq, [217](#)
  - l4\_factory\_create\_irq\_u, [218](#)
  - l4\_factory\_create\_task, [219](#)
  - l4\_factory\_create\_task\_u, [220](#)
  - l4\_factory\_create\_thread, [221](#)
  - l4\_factory\_create\_thread\_u, [222](#)
  - l4\_factory\_create\_vm, [223](#)
  - l4\_factory\_create\_vm\_u, [224](#)
- factory
  - L4Re::Env, [1311](#), [1312](#)
- failed
  - L4virtio::Svr::Dev\_status, [1644](#), [1645](#)
- failed\_bfm\_t
  - L4virtio::Svr::Dev\_status, [1642](#)
- fchmod
  - L4Re::Vfs::Generic\_file, [1463](#)
- fd
  - l4\_vhw\_entry, [1278](#)
- fdatasync
  - L4Re::Vfs::Regular\_file, [1472](#)
- features
  - l4\_icu\_info\_t, [1248](#)

- features\_ok
  - L4virtio::Svr::Dev\_status, 1645
- features\_ok\_bfm\_t
  - L4virtio::Svr::Dev\_status, 1642
- Fiasco extensions, 225
  - fiasco\_gdt\_get\_entry\_offset, 227
  - fiasco\_gdt\_set, 227
  - fiasco\_ldt\_set, 228
  - fiasco\_tbuf\_get\_status, 228
  - fiasco\_tbuf\_get\_status\_phys, 228
  - fiasco\_tbuf\_log, 228
  - fiasco\_tbuf\_log\_3val, 229
  - fiasco\_tbuf\_log\_binary, 230
- Fiasco real time scheduling extensions, 231
- Fiasco-UX Virtual devices, 232
  - l4\_vhw\_entry\_type, 232
- fiasco\_amd64\_segment\_info
  - amd64/l4/sys/segment.h, 1763
- fiasco\_amd64\_set\_fs
  - amd64/l4/sys/segment.h, 1764
  - amd64/l4f/l4/sys/segment.h, 1759
- fiasco\_amd64\_set\_segment\_base
  - amd64/l4/sys/segment.h, 1764
  - amd64/l4f/l4/sys/segment.h, 1760
- fiasco\_gdt\_get\_entry\_offset
  - Fiasco extensions, 227
- fiasco\_gdt\_set
  - Fiasco extensions, 227
- fiasco\_ldt\_set
  - Fiasco extensions, 228
- fiasco\_tbuf\_get\_status
  - Fiasco extensions, 228
- fiasco\_tbuf\_get\_status\_phys
  - Fiasco extensions, 228
- fiasco\_tbuf\_log
  - Fiasco extensions, 228
- fiasco\_tbuf\_log\_3val
  - Fiasco extensions, 229
- fiasco\_tbuf\_log\_binary
  - Fiasco extensions, 230
- finalize\_request
  - L4virtio::Svr::Block\_dev, 1614
- find
  - cxx::Bits::Bst, 779
  - cxx::String, 851
  - L4::Basic\_registry, 877
  - L4Re::Rm, 1367
  - L4virtio::Svr::Driver\_mem\_list\_t, 1655
- find\_next\_used
  - L4virtio::Driver::Virtqueue, 1602
- find\_node
  - cxx::Bits::Base\_avl\_set, 763
  - cxx::Bits::Bst, 780
- first
  - L4::Kip::Mem\_desc, 1085
- first\_free\_cap
  - L4Re::Env, 1312
- first\_free\_utcb
  - L4Re::Env, 1313
- Flags
  - L4::Types::Flags, 1223
  - L4Re::Video::Goos, 1484
  - L4Re::Video::View, 1499
- flags
  - l4\_exc\_regs\_t, 1245
  - l4\_msgtag\_t, 1255
  - L4Re::Dataspace, 1295
  - l4re\_env\_cap\_entry\_t, 1510
  - L4virtio::Svr::Driver\_mem\_region\_t, 1664
- Flex pages, 234
  - L4\_cap\_fpage\_rights, 236
  - l4\_fpage, 240
  - l4\_fpage\_all, 240
  - l4\_fpage\_cacheability\_opt\_t, 238
  - l4\_fpage\_consts, 238
  - l4\_fpage\_contains, 241
  - l4\_fpage\_invalid, 241
  - l4\_fpage\_ioport, 242
  - l4\_fpage\_max\_order, 242
  - l4\_fpage\_memaddr, 243
  - l4\_fpage\_obj, 244
  - l4\_fpage\_page, 245
  - L4\_fpage\_rights, 239
  - l4\_fpage\_rights, 245
  - l4\_fpage\_set\_rights, 246
  - l4\_fpage\_size, 246
  - l4\_fpage\_type, 247
  - l4\_iofpage, 247
  - l4\_is\_fpage\_writable, 248
  - l4\_obj\_fpage, 248
  - L4\_obj\_fpage\_ctl, 239
- foreach\_available\_event
  - L4Re::Util::Event\_buffer\_consumer\_t, 1404
- fpage
  - L4::Cap\_base, 891
- free
  - cxx::Base\_slab\_static, 721
  - cxx::List\_alloc, 811
  - cxx::Slab, 842
  - L4Re::Cap\_alloc, 1287
  - L4Re::Mem\_alloc, 1341
  - L4Re::Util::Counting\_cap\_alloc, 1388
- free\_area
  - L4Re::Rm, 1368
- free\_capability
  - L4Re::Util::Names::Name\_space, 1419
- free\_descriptor
  - L4virtio::Driver::Virtqueue, 1602
- free\_dynamic\_entry
  - L4Re::Util::Names::Name\_space, 1420
- free\_epiface
  - L4Re::Util::Names::Name\_space, 1420
- free\_fd
  - L4Re::Vfs::Fs, 1459
- free\_objects
  - cxx::Base\_slab, 717

- cxx::Base\_slab\_static, [722](#)
- from\_ci
  - L4::lpc::Cap, [965](#)
- from\_dec
  - cxx::String, [852](#)
- from\_hex
  - cxx::String, [853](#)
- from\_raw
  - L4::Types::Flags, [1224](#)
- fstat64
  - L4Re::Vfs::Be\_file, [1443](#)
  - L4Re::Vfs::Generic\_file, [1464](#)
- fsync
  - L4Re::Vfs::Regular\_file, [1472](#)
- ftruncate64
  - L4Re::Vfs::Regular\_file, [1472](#)
- full
  - L4virtio::Svr::Driver\_mem\_list\_t, [1655](#)
- Functions to manipulate the local IDT, [250](#)
- g
  - L4Re::Video::Pixel\_info, [1495](#)
- get
  - cxx::Auto\_ptr, [697](#)
  - cxx::Bitfield, [726](#)
  - cxx::Ref\_ptr, [834](#)
  - L4::lpc::Istream, [980](#), [981](#)
  - L4Re::Env, [1313](#)
  - L4Re::Video::Color\_component, [1479](#)
  - L4vbus::Gpio\_module, [1544](#)
  - L4vbus::Gpio\_pin, [1552](#)
  - L4virtio::Ptr, [1606](#)
- get\_attr
  - L4::Vcon, [1234](#)
- get\_avail\_idx
  - L4virtio::Virtqueue, [1687](#)
- get\_buffer
  - L4Re::Event, [1324](#)
- get\_buffer\_demand
  - L4::Server\_object\_t, [1157](#)
- get\_cap
  - L4Re::Env, [1314](#), [1315](#)
- get\_cap\_alloc
  - L4Re::Cap\_alloc, [1288](#)
- get\_cmd
  - L4virtio::Svr::Dev\_config, [1629](#)
- get\_epiface
  - L4Re::Util::Names::Name\_space, [1420](#)
- get\_fb
  - L4Re::Util::Video::Goos\_svr, [1437](#)
- get\_file
  - L4Re::Vfs::Fs, [1459](#)
- get\_infos
  - L4::lpc\_gate, [1031](#)
- get\_lock
  - L4Re::Vfs::Regular\_file, [1473](#)
- get\_next\_item
  - cxx::List\_item, [814](#)
- get\_object\_name
  - L4::Debugger, [902](#)
- get\_prev\_item
  - cxx::List\_item, [814](#)
- get\_rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1048](#)
  - L4Re::Util::Br\_manager, [1377](#)
- get\_resource
  - L4vbus::Device, [1537](#)
- get\_static\_buffer
  - L4Re::Video::Goos, [1486](#)
- get\_status\_flags
  - L4Re::Vfs::Generic\_file, [1464](#)
- get\_tail\_avail\_idx
  - L4virtio::Virtqueue, [1687](#)
- get\_unshifted
  - cxx::Bitfield, [727](#)
- get\_value
  - L4::lpc::Varg, [1021](#)
- get\_writeback
  - L4virtio::Svr::Block\_dev, [1614](#)
- global\_id
  - L4::Debugger, [903](#)
- gran\_offset
  - l4\_sched\_cpu\_set\_t, [1257](#)
- granularity
  - l4\_sched\_cpu\_set\_t, [1256](#)
- guest\_features
  - L4virtio::Svr::Dev\_config, [1630](#)
- H\_list\_item
  - cxx, [639](#)
- H\_list\_item\_t
  - cxx::H\_list\_item\_t, [800](#)
- handle\_expired\_timeouts
  - L4::lpc\_svr::Timeout\_queue, [1057](#)
- handle\_mem\_cmd\_write
  - L4virtio::Svr::Device\_t, [1650](#)
- has\_alpha
  - L4Re::Video::Pixel\_info, [1496](#)
- has\_more
  - L4virtio::Svr::Request\_processor, [1667](#)
- hdr
  - L4virtio::Svr::Dev\_config, [1631](#)
- i
  - L4vcpu::Vcpu, [1586](#)
- IA32 Port I/O API, [251](#)
- l4util\_in16, [251](#)
- l4util\_in32, [253](#)
- l4util\_in8, [253](#)
- l4util\_ins16, [254](#)
- l4util\_ins32, [254](#)
- l4util\_ins8, [255](#)
- l4util\_out16, [255](#)
- l4util\_out32, [255](#)
- l4util\_out8, [256](#)
- l4util\_outs16, [256](#)
- l4util\_outs32, [257](#)
- l4util\_outs8, [257](#)

- IO interface, [258](#)
  - [l4io\\_device\\_types\\_t](#), [259](#)
  - [l4io\\_has\\_resource](#), [260](#)
  - [l4io\\_iomem\\_flags\\_t](#), [259](#)
  - [l4io\\_lookup\\_device](#), [261](#)
  - [l4io\\_lookup\\_resource](#), [261](#)
  - [l4io\\_release\\_iomem](#), [262](#)
  - [l4io\\_release\\_ioport](#), [262](#)
  - [l4io\\_request\\_iomem](#), [262](#)
  - [l4io\\_request\\_iomem\\_region](#), [263](#)
  - [l4io\\_request\\_ioport](#), [264](#)
  - [l4io\\_request\\_resource\\_iomem](#), [264](#)
  - [l4io\\_resource\\_t](#), [259](#)
  - [l4io\\_resource\\_types\\_t](#), [260](#)
  - [l4io\\_search\\_iomem\\_region](#), [265](#)
- IPC-Gate API, [266](#)
  - [l4\\_ipc\\_gate\\_bind\\_thread](#), [267](#)
  - [l4\\_ipc\\_gate\\_get\\_infos](#), [267](#)
  - [l4\\_rcv\\_ep\\_bind\\_thread](#), [268](#)
- IRQ handling library, [269](#)
- IRQs, [270](#)
  - [l4\\_irq\\_attach](#), [272](#)
  - [l4\\_irq\\_attach\\_u](#), [272](#)
  - [l4\\_irq\\_detach](#), [273](#)
  - [l4\\_irq\\_detach\\_u](#), [274](#)
  - [L4\\_irq\\_mode](#), [271](#)
  - [l4\\_irq\\_mux\\_chain](#), [275](#)
  - [l4\\_irq\\_mux\\_chain\\_u](#), [276](#)
  - [l4\\_irq\\_receive](#), [277](#)
  - [l4\\_irq\\_receive\\_u](#), [278](#)
  - [l4\\_irq\\_trigger](#), [279](#)
  - [l4\\_irq\\_trigger\\_u](#), [280](#)
  - [l4\\_irq\\_unmask](#), [281](#)
  - [l4\\_irq\\_unmask\\_u](#), [281](#)
  - [l4\\_irq\\_wait](#), [282](#)
  - [l4\\_irq\\_wait\\_u](#), [283](#)
- id
  - [L4::Kobject\\_typeid](#), [1103](#)
- id\_received
  - [L4::lpc::Gen\\_fpage](#), [967](#)
- idle\_time
  - [L4::Scheduler](#), [1141](#)
- indirect
  - [L4virtio::Virtqueue::Desc::Flags](#), [1701](#)
- indirect\_bfm\_t
  - [L4virtio::Virtqueue::Desc::Flags](#), [1700](#)
- info
  - [L4::lcu](#), [934](#)
  - [L4::Scheduler](#), [1142](#)
  - [L4Re::Dataspace](#), [1297](#)
  - [L4Re::Video::Goos](#), [1487](#)
  - [L4Re::Video::View](#), [1500](#)
- Info\_sub\_type
  - [L4::Kip::Mem\\_desc](#), [1081](#)
- init
  - [L4Re::Util::Event\\_t](#), [1413](#)
  - [L4virtio::Svr::Driver\\_mem\\_list\\_t](#), [1656](#)
- init\_infos
  - [L4Re::Util::Video::Goos\\_svr](#), [1437](#)
- init\_mem\_info
  - [L4virtio::Svr::Device\\_t](#), [1650](#)
- init\_poll
  - [L4Re::Util::Event\\_t](#), [1414](#)
- init\_queue
  - [L4virtio::Driver::Virtqueue](#), [1603](#)
- Initial Environment, [285](#)
  - [l4re\\_env](#), [286](#)
  - [l4re\\_env\\_get\\_cap](#), [286](#)
  - [l4re\\_env\\_get\\_cap\\_e](#), [287](#)
  - [l4re\\_env\\_get\\_cap\\_l](#), [288](#)
  - [l4re\\_kip](#), [289](#)
- initial\_caps
  - [L4Re::Env](#), [1315](#)
- initialize\_rings
  - [L4virtio::Driver::Virtqueue](#), [1604](#)
- insert
  - [cxx::Avl\\_map](#), [703](#)
  - [cxx::Avl\\_tree](#), [711](#)
  - [cxx::Bits::Base\\_avl\\_set](#), [764](#)
  - [cxx::H\\_list](#), [796](#)
- insert\_after
  - [cxx::H\\_list](#), [796](#)
- insert\_before
  - [cxx::H\\_list](#), [797](#)
- insert\_next\_item
  - [cxx::List\\_item](#), [814](#)
- insert\_prev\_item
  - [cxx::List\\_item](#), [815](#)
- Integer Types, [290](#)
  - [l4\\_int16\\_t](#), [291](#)
  - [l4\\_int32\\_t](#), [291](#)
  - [l4\\_int64\\_t](#), [292](#)
  - [l4\\_int8\\_t](#), [292](#)
  - [l4\\_uint16\\_t](#), [292](#)
  - [l4\\_uint32\\_t](#), [292](#)
  - [l4\\_uint64\\_t](#), [292](#)
  - [l4\\_uint8\\_t](#), [292](#)
- interface
  - [L4::Meta](#), [1110](#)
- Interface for asynchronous ISR handlers with a given I↔RQ capability., [293](#)
  - [l4irq\\_request\\_cap](#), [293](#)
- Interface for asynchronous ISR handlers., [295](#)
  - [l4irq\\_release](#), [295](#)
  - [l4irq\\_request](#), [296](#)
- Interface using direct functionality., [297](#), [303](#)
  - [l4irq\\_attach](#), [297](#)
  - [l4irq\\_attach\\_cap](#), [303](#)
  - [l4irq\\_attach\\_cap\\_ft](#), [303](#)
  - [l4irq\\_attach\\_ft](#), [298](#)
  - [l4irq\\_attach\\_thread](#), [298](#)
  - [l4irq\\_attach\\_thread\\_cap](#), [304](#)
  - [l4irq\\_attach\\_thread\\_cap\\_ft](#), [304](#)
  - [l4irq\\_attach\\_thread\\_ft](#), [299](#)
  - [l4irq\\_detach](#), [299](#)
  - [l4irq\\_unmask](#), [300](#)



- l4irq\_unmask\_and\_wait\_any, 300
- l4irq\_wait, 300
- l4irq\_wait\_any, 302
- Internal constants, 307
- Internal functions, 309
- Internal Helpers, 306
- internal\_loop
  - L4::Server, 1151
- Interrupt controller, 310
  - l4\_icu\_bind, 312
  - l4\_icu\_bind\_u, 313
  - L4\_icu\_flags, 312
  - l4\_icu\_info, 314
  - l4\_icu\_info\_t, 311
  - l4\_icu\_info\_u, 314
  - l4\_icu\_mask, 315
  - l4\_icu\_mask\_u, 316
  - l4\_icu\_msi\_info, 317
  - l4\_icu\_msi\_info\_u, 318
  - l4\_icu\_set\_mode, 318
  - l4\_icu\_set\_mode\_u, 319
  - l4\_icu\_unbind, 320
  - l4\_icu\_unbind\_u, 321
  - l4\_icu\_unmask, 322
  - l4\_icu\_unmask\_u, 322
- Invalid\_capability
  - L4::Invalid\_capability, 943
- Invalid\_type
  - L4virtio::Ptr, 1606
- io.h
  - l4io\_get\_root\_device, 1817
  - l4io\_iterate\_devices, 1817
  - l4io\_request\_all\_ioports, 1818
  - l4io\_request\_icu, 1818
- io\_page\_fault
  - L4::io\_pager, 946
- ioctl
  - L4Re::Vfs::Special\_file, 1477
- lostream
  - L4::lpc::lostream, 972
- ipc\_client
  - L4\_RPC\_DEF, 2085
- ipc\_iface
  - L4\_INLINE\_RPC\_NF\_OP, 2089
  - L4\_INLINE\_RPC\_NF, 2089
  - L4\_INLINE\_RPC\_OP, 2090
  - L4\_INLINE\_RPC, 2088
  - L4\_RPC\_NF\_OP, 2091
  - L4\_RPC\_NF, 2091
  - L4\_RPC\_OP, 2092
  - L4\_RPC, 2090
- ipc\_stream
  - operator<<, 1864–1866
  - operator>>, 1867–1871
- irq
  - L4Re::Util::Event\_t, 1415
- irq\_disable\_save
  - L4vcpu::Vcpu, 1586
- irq\_enable
  - L4vbus::Pci\_dev, 1561
  - L4vbus::Pci\_host\_bridge, 1567
  - L4vcpu::Vcpu, 1587
- irq\_no
  - l4\_vhw\_entry, 1278
- irq\_restore
  - L4vcpu::Vcpu, 1588
- is\_compatible
  - L4vbus::Device, 1537
- is\_compound
  - L4::lpc::Gen\_fpage, 968
- is\_irq\_entry
  - L4vcpu::Vcpu, 1588
- is\_nil
  - L4::lpc::Varg, 1022
- is\_of
  - L4::lpc::Varg, 1022
- is\_of\_int
  - L4::lpc::Varg, 1022
- is\_online
  - L4::Scheduler, 1142
- is\_page\_fault\_entry
  - L4vcpu::Vcpu, 1589
- is\_static
  - L4Re::Util::Dataspace\_svr, 1395
- is\_valid
  - L4::Cap\_base, 892
  - L4virtio::Ptr, 1607
- is\_virtual
  - L4::Kip::Mem\_desc, 1086
- is\_writable
  - L4virtio::Svr::Driver\_mem\_region\_t, 1664
- Istream
  - L4::lpc::Istream, 979
- items
  - cxx::List, 805
- iter
  - cxx::Bits::Basic\_list, 770
  - cxx::H\_list, 797
- Kernel Debugger, 324
  - l4\_debugger\_global\_id, 324
  - l4\_debugger\_kobj\_to\_id, 325
  - l4\_debugger\_set\_object\_name, 325
- Kernel Interface Page, 327
  - l4\_kernel\_info\_version\_offset, 328
  - l4\_kip\_clock, 329
  - l4\_kip\_clock\_lw, 330
  - l4\_kip\_version, 330
  - l4\_kip\_version\_string, 331
- Kernel Interface Page API, 332
  - l4util\_kip\_for\_each\_feature, 332
  - l4util\_kip\_kernel\_abi\_version, 333
  - l4util\_kip\_kernel\_has\_feature, 333
  - l4util\_kip\_kernel\_is\_ux, 333
  - l4util\_memdesc\_vm\_high, 333
- Kernel Objects, 335
- kobj\_to\_id

- L4::Debugger, 903
- kobject\_typeid
  - L4 kernel object type information, 367
- Kumem allocator utility, 337
- Kumem utilities, 338
  - kumem\_alloc, 338
- kumem\_alloc
  - Kumem utilities, 338
- kumem\_alloc.h
  - l4re\_util\_kumem\_alloc, 1932
- L4, 640
  - cap\_cast, 644, 645
  - cap\_dynamic\_cast, 646
  - cap\_reinterpret\_cast, 647, 648
  - throw\_ipc\_exception, 649, 650
- L4 IPC Opcodes, 340
  - L4\_icu\_opcode, 340
  - L4\_ipc\_gate\_ops, 341
  - L4\_platform\_ctl\_ops, 342
  - L4\_task\_ops, 342
  - L4\_thread\_ops, 342
  - L4\_vcon\_ops, 343
- L4 kernel object type information, 367
  - kobject\_typeid, 367
- L4 VIRTIO Block Device, 344
  - L4virtio\_block\_operations, 344
  - L4virtio\_block\_status, 345
- L4 VIRTIO Interface, 346
- L4 VIRTIO Transport Layer, 347
  - L4\_virtio\_cmd, 349
  - L4\_virtio\_irq\_status, 349
  - L4\_virtio\_opcodes, 350
  - l4virtio\_config\_queue, 351
  - l4virtio\_config\_queue\_t, 349
  - l4virtio\_config\_queues, 352
  - l4virtio\_device\_config, 353
  - l4virtio\_device\_config\_ds, 353
  - l4virtio\_device\_ids, 350
  - l4virtio\_device\_notification\_irq, 354
  - l4virtio\_device\_status, 351
  - l4virtio\_feature\_bits, 351
  - l4virtio\_register\_ds, 354
  - l4virtio\_register\_iface, 355
  - l4virtio\_set\_status, 356
- L4 Vbus functions, 357
  - l4vbus\_assign\_dma\_domain, 358
  - l4vbus\_dma\_domain\_assign\_flags, 358
  - l4vbus\_get\_device, 359
  - l4vbus\_get\_device\_by\_hid, 360
  - l4vbus\_get\_hid, 361
  - l4vbus\_get\_next\_device, 361
  - l4vbus\_get\_resource, 362
  - l4vbus\_is\_compatible, 363
  - l4vbus\_release\_resource, 363
  - l4vbus\_request\_resource, 364
  - l4vbus\_vicu\_get\_cap, 365
- l4/cxx/alloc.h, 2248
- l4/cxx/atomic.h, 2262, 2263
- l4/cxx/avl\_map, 1825, 1826
- l4/cxx/avl\_set, 1828, 1829
- l4/cxx/avl\_tree, 1832, 1834
- l4/cxx/basic\_ostream, 1838, 1839
- l4/cxx/basic\_vector.h, 1842, 1843
- l4/cxx/bits/bst.h, 1843, 1845
- l4/cxx/bits/bst\_base.h, 1847, 1849
- l4/cxx/bits/bst\_iter.h, 1850, 1852
- l4/cxx/exceptions, 1853, 1855
- l4/cxx/iostream, 1858, 1859
- l4/cxx/ipc\_helper, 1859, 1861
- l4/cxx/ipc\_server, 2098, 2099
- l4/cxx/ipc\_stream, 1861, 1872
- l4/cxx/l4iostream, 1882, 1883
- l4/cxx/l4types.h, 1883, 1884
- l4/cxx/main\_thread, 1885, 1886
- l4/cxx/pair, 1886, 1888
- l4/cxx/std\_exc\_io, 1888, 1889
- l4/cxx/string.h, 1890, 1892
- l4/cxx/thread, 2217, 2218
- l4/irq/irq.h, 1892, 1894
- l4/libedid/edid.h, 1906, 1907
- l4/re/c/dataspace.h, 1908, 1909
- l4/re/c/debug.h, 1910, 1911
- l4/re/c/dma\_space.h, 1912, 1914
- l4/re/c/event.h, 1915, 1917
- l4/re/c/log.h, 1919, 1920
- l4/re/c/mem\_alloc.h, 1921, 1923
- l4/re/c/namespace.h, 1924, 1925
- l4/re/c/rm.h, 1926, 1927
- l4/re/c/util/cap\_alloc.h, 1930, 1931
- l4/re/c/util/kumem\_alloc.h, 1931, 1933
- l4/re/c/util/video/goos\_fb.h, 1933, 1934
- l4/re/c/video/colors.h, 1935, 1937
- l4/re/c/video/goos.h, 1937, 1939
- l4/re/c/video/view.h, 1940, 1943
- l4/re/cap\_alloc, 1944, 1945
- l4/re/consts, 1951, 1952
- l4/re/consts.h, 2082, 2083
- l4/re/dataspace, 1953, 1954
- l4/re/dataspace-sys.h, 1955, 1956
- l4/re/debug, 1956, 1958
- l4/re/dma\_space, 1958, 1960
- l4/re/elf\_aux.h, 1961, 1962
- l4/re/env, 1963, 1964
- l4/re/env.h, 1965, 1967
- l4/re/error\_helper, 1969, 1971
- l4/re/event.h, 1917, 1918
- l4/re/impl/dataspace\_impl.h, 1975, 1976
- l4/re/impl/mem\_alloc\_impl.h, 1977, 1978
- l4/re/impl/namespace\_impl.h, 1979, 1980
- l4/re/impl/rm\_impl.h, 1981, 1982
- l4/re/l4aux.h, 1984, 1985
- l4/re/log, 1985, 1987
- l4/re/log-sys.h, 1987, 1988
- l4/re/mem\_alloc, 1988, 1990
- l4/re/mem\_alloc-sys.h, 1990, 1991
- l4/re/namespace, 1992, 1993

- [l4/re/namespace-sys.h](#), [1994](#), [1995](#)
- [l4/re/parent](#), [1995](#), [1997](#)
- [l4/re/parent-sys.h](#), [1997](#), [1998](#)
- [l4/re/protocols.h](#), [1998](#), [1999](#)
- [l4/re/rm](#), [2000](#), [2001](#)
- [l4/re/rm-sys.h](#), [2005](#), [2006](#)
- [l4/re/shared\\_cap](#), [2006](#), [2008](#)
- [l4/re/unique\\_cap](#), [2011](#), [2013](#)
- [l4/re/util/bitmap\\_cap\\_alloc](#), [2016](#), [2017](#)
- [l4/re/util/cap](#), [2018](#), [2020](#)
- [l4/re/util/cap\\_alloc](#), [1947](#), [1949](#)
- [l4/re/util/cap\\_alloc\\_impl.h](#), [2020](#), [2021](#)
- [l4/re/util/counting\\_cap\\_alloc](#), [2022](#), [2023](#)
- [l4/re/util/event](#), [1972](#), [1973](#)
- [l4/re/util/item\\_alloc](#), [2025](#), [2026](#)
- [l4/re/util/kumem\\_alloc](#), [2027](#), [2028](#)
- [l4/re/util/region\\_mapping](#), [2029](#), [2030](#)
- [l4/re/util/shared\\_cap](#), [2009](#), [2010](#)
- [l4/re/util/unique\\_cap](#), [2013](#), [2015](#)
- [l4/re/video/goos-sys.h](#), [2035](#), [2036](#)
- [l4/shmc/shmc.h](#), [2036](#), [2039](#)
- [l4/sigma0/sigma0.h](#), [2041](#), [2043](#)
- [l4/sys/\\_\\_kernel\\_object\\_impl.h](#), [2044](#), [2045](#)
- [l4/sys/\\_\\_ktrace\\_impl.h](#), [2046](#), [2047](#)
- [l4/sys/\\_\\_typeinfo.h](#), [2048](#), [2051](#)
- [l4/sys/assert.h](#), [2251](#), [2253](#)
- [l4/sys/cache.h](#), [2061](#), [2062](#)
- [l4/sys/capability](#), [2068](#), [2070](#)
- [l4/sys/compiler.h](#), [2072](#), [2073](#)
- [l4/sys/consts.h](#), [2075](#), [2077](#)
- [l4/sys/cxx/ipc\\_client](#), [2083](#), [2085](#)
- [l4/sys/cxx/ipc\\_iface](#), [2086](#), [2093](#)
- [l4/sys/cxx/ipc\\_types](#), [2100](#), [2102](#)
- [l4/sys/cxx/smart\\_capability\\_1x](#), [2109](#), [2110](#)
- [l4/sys/cxx/types](#), [2113](#), [2114](#)
- [l4/sys/debugger](#), [2115](#), [2116](#)
- [l4/sys/debugger.h](#), [2117](#), [2121](#)
- [l4/sys/err.h](#), [2124](#), [2126](#)
- [l4/sys/exception](#), [2126](#), [2128](#)
- [l4/sys/factory](#), [2128](#), [2130](#)
- [l4/sys/factory.h](#), [2132](#), [2133](#)
- [l4/sys/icu](#), [2138](#), [2139](#)
- [l4/sys/icu.h](#), [2139](#), [2142](#)
- [l4/sys/ipc.h](#), [2145](#), [2148](#)
  - [l4\\_ipc\\_to\\_errno](#), [2147](#)
- [l4/sys/ipc\\_gate](#), [2154](#), [2156](#)
- [l4/sys/ipc\\_gate.h](#), [2156](#), [2158](#)
- [l4/sys/irq](#), [2159](#), [2161](#)
- [l4/sys/irq.h](#), [1901](#), [1903](#)
- [l4/sys/kernel\\_object.h](#), [2163](#), [2164](#)
- [l4/sys/kip](#), [2165](#), [2167](#)
- [l4/sys/kip.h](#), [2299](#), [2300](#)
- [l4/sys/ktrace.h](#), [2168](#), [2170](#)
- [l4/sys/l4int.h](#), [2171](#), [2172](#)
- [l4/sys/memdesc.h](#), [2175](#), [2177](#)
- [l4/sys/meta](#), [2179](#), [2181](#)
- [l4/sys/pager](#), [2181](#), [2183](#)
- [l4/sys/platform\\_control](#), [2183](#), [2185](#)
- [l4/sys/platform\\_control.h](#), [2185](#), [2187](#)
- [l4/sys/rcv\\_endpoint](#), [2189](#), [2190](#)
- [l4/sys/rcv\\_endpoint.h](#), [2191](#), [2193](#)
- [l4/sys/scheduler](#), [2193](#), [2195](#)
- [l4/sys/scheduler.h](#), [2195](#), [2197](#)
- [l4/sys/semaphore](#), [2200](#), [2202](#)
- [l4/sys/smart\\_capability](#), [2202](#), [2204](#)
- [l4/sys/task](#), [2206](#), [2208](#)
- [l4/sys/task.h](#), [2209](#), [2210](#)
- [l4/sys/thread](#), [2214](#), [2215](#)
- [l4/sys/thread.h](#), [2329](#), [2331](#)
- [l4/sys/typeinfo\\_svr](#), [2219](#), [2221](#)
- [l4/sys/types.h](#), [1819](#), [1822](#)
  - [l4\\_capability\\_next](#), [1822](#)
- [l4/sys/utcb.h](#), [2221](#), [2223](#)
- [l4/sys/vcon](#), [2234](#), [2236](#)
- [l4/sys/vcon.h](#), [2236](#), [2239](#)
- [l4/sys/vhw.h](#), [2242](#), [2244](#)
- [l4/sys/vm](#), [2245](#), [2246](#)
- [l4/util/alloc.h](#), [2246](#), [2247](#)
- [l4/util/assert.h](#), [2249](#), [2250](#)
- [l4/util/atomic.h](#), [2254](#), [2257](#)
- [l4/util/backtrace.h](#), [2263](#), [2265](#)
- [l4/util/base64.h](#), [2265](#), [2267](#)
- [l4/util/bitops.h](#), [2267](#), [2269](#)
- [l4/util/elf.h](#), [2272](#), [2285](#)
- [l4/util/getopt.h](#), [2295](#)
- [l4/util/keymap.h](#), [2296](#), [2297](#)
- [l4/util/kip.h](#), [2297](#), [2298](#)
- [l4/util/kprintf.h](#), [2302](#), [2303](#)
- [l4/util/l4\\_macros.h](#), [1805](#), [1806](#)
- [l4/util/list\\_alloc.h](#), [2303](#), [2306](#)
- [l4/util/lock.h](#), [2306](#), [2307](#)
- [l4/util/mb\\_info.h](#), [2308](#), [2311](#)
- [l4/util/parse\\_cmd.h](#), [2315](#), [2316](#)
- [l4/util/rand.h](#), [2316](#), [2317](#)
- [l4/util/reboot.h](#), [2318](#)
- [l4/util/sll.h](#), [2319](#)
- [l4/util/splitlog2.h](#), [2323](#), [2324](#)
- [l4/util/stack.h](#), [2324](#), [2326](#)
- [l4/util/thread.h](#), [2326](#), [2328](#)
  - [\\_\\_L4UTIL\\_THREAD\\_FUNC](#), [2328](#)
- [l4/vbus/vbus.h](#), [2340](#), [2342](#)
- [l4/vbus/vbus\\_interfaces.h](#), [2343](#), [2345](#)
- [l4/vbus/vbus\\_types.h](#), [2345](#), [2347](#)
- [L4::Alloc\\_list](#), [872](#)
- [L4::Arm\\_smccc](#), [873](#)
  - [call](#), [873](#)
- [L4::Base\\_exception](#), [874](#)
- [L4::Basic\\_registry](#), [876](#)
  - [dispatch](#), [877](#)
  - [find](#), [877](#)
- [L4::Bounds\\_error](#), [878](#)
- [L4::Cap](#)
  - [Cap](#), [883](#), [884](#)
  - [copy](#), [885](#)
  - [move](#), [885](#)
- [L4::Cap< T >](#), [881](#)

- L4::Cap\_base, 886
  - \_c, 896
  - cap, 889
  - Cap\_base, 888, 889
  - Cap\_type, 888
  - copy, 890
  - fpage, 891
  - is\_valid, 892
  - move, 893
  - No\_init\_type, 888
  - snd\_base, 894
  - validate, 895
- L4::Com\_error, 897
  - Com\_error, 899
- L4::Debugger, 899
  - get\_object\_name, 902
  - global\_id, 903
  - kobj\_to\_id, 903
  - query\_log\_name, 903
  - query\_log\_typeid, 904
  - set\_object\_name, 905
  - switch\_log, 905
- L4::Element\_already\_exists, 906
- L4::Element\_not\_found, 908
- L4::Exception, 910
  - Rpcs, 910
- L4::Exception\_tracer, 911
- L4::Factory, 913
  - create, 916, 917
  - create\_factory, 918
  - create\_gate, 919
  - create\_irq, 920
  - create\_task, 921
  - create\_thread, 922
  - create\_vm, 923
- L4::Factory::Lstr, 924
  - Lstr, 925
- L4::Factory::Nil, 925
- L4::Factory::S, 926
  - operator l4\_msgtag\_t, 928
  - operator<<, 928–930
  - S, 927
- L4::IOModifier, 949
- L4::lcu, 931
  - bind, 933
  - info, 934
  - mask, 936
  - msi\_info, 937
  - set\_mode, 937
  - unbind, 938
- L4::lcu::Info, 939
- L4::Invalid\_capability, 940
  - cap, 943
  - Invalid\_capability, 943
- L4::io\_pager, 944
  - io\_page\_fault, 946
- L4::lommu, 946
  - bind, 949
  - unbind, 949
- L4::lpc, 651
  - buf\_cp\_in, 653
  - buf\_cp\_out, 654
  - buf\_in, 654
  - make\_cap, 655
  - make\_cap\_full, 656
  - make\_cap\_rw, 657
  - make\_cap\_rws, 658
  - msg\_ptr, 658
  - read, 659
  - str\_cp\_in, 659
- L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >, 950
- L4::lpc::Array\_in\_buf< ELEM\_TYPE, LEN\_TYPE, MAX >, 953
- L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE >, 954
- L4::lpc::As\_value< T >, 956
- L4::lpc::Buf\_item, 957
- L4::lpc::Call, 958
- L4::lpc::Call\_t< RIGHTS >, 959
- L4::lpc::Call\_zero\_send\_timeout, 961
- L4::lpc::Cap
  - Cap, 964
  - from\_ci, 965
- L4::lpc::Cap< T >, 962
- L4::lpc::Gen\_fpage
  - cap\_received, 967
  - id\_received, 967
  - is\_compound, 968
  - local\_id\_received, 968
- L4::lpc::Gen\_fpage< T >, 965
- L4::lpc::In\_out< T >, 969
- L4::lpc::Iostream, 969
  - call, 973
  - loststream, 972
  - reply\_and\_wait, 974
  - reset, 975
- L4::lpc::Istream, 976
  - get, 980, 981
  - Istream, 979
  - receive, 982
  - reset, 983
  - skip, 983
  - tag, 984
  - wait, 985
- L4::lpc::Msg, 660
  - align\_to, 662
  - check\_size, 663, 664
  - msg\_add, 664
  - msg\_get, 665
- L4::lpc::Msg::Cnt\_val\_ops< MTYPE, DIR, CLASS >, 986
- L4::lpc::Msg::Cls\_buffer, 987
- L4::lpc::Msg::Cls\_data, 989
- L4::lpc::Msg::Cls\_item, 990
- L4::lpc::Msg::Dir\_in, 991
- L4::lpc::Msg::Dir\_out, 992
- L4::lpc::Msg::Do\_in\_data, 993

- L4::lpc::Msg::Do\_in\_items, [994](#)
- L4::lpc::Msg::Do\_out\_data, [995](#)
- L4::lpc::Msg::Do\_out\_items, [996](#)
- L4::lpc::Msg::Do\_rcv\_buffers, [997](#)
- L4::lpc::Msg::Elem< Array< A, LEN > >, [1000](#)
- L4::lpc::Msg::Elem< Array< A, LEN > &>, [999](#)
- L4::lpc::Msg::Elem< Array\_ref< A, LEN > &>, [1001](#)
- L4::lpc::Msg::Is\_valid\_rpc\_type< T >, [1002](#)
- L4::lpc::Msg::Svr\_arg\_pack< IPC\_TYPE >, [1004](#)
- L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >, [1004](#)
- L4::lpc::Msg\_ptr
  - Msg\_ptr, [1006](#)
- L4::lpc::Msg\_ptr< T >, [1005](#)
- L4::lpc::Opt< T >, [1006](#)
- L4::lpc::Ostream, [1008](#)
  - put, [1011](#), [1012](#)
  - send, [1012](#)
  - tag, [1013](#)
- L4::lpc::Out< T >, [1014](#)
- L4::lpc::Ret\_array< T >, [1015](#)
- L4::lpc::Send\_only, [1016](#)
- L4::lpc::Small\_buf, [1016](#)
  - Small\_buf, [1017](#)
- L4::lpc::Snd\_item, [1018](#)
- L4::lpc::Str\_cp\_in
  - Str\_cp\_in, [1019](#)
- L4::lpc::Str\_cp\_in< T >, [1018](#)
- L4::lpc::Varg, [1020](#)
  - data, [1021](#)
  - get\_value, [1021](#)
  - is\_nil, [1022](#)
  - is\_of, [1022](#)
  - is\_of\_int, [1022](#)
  - length, [1022](#)
  - tag, [1023](#)
  - type, [1024](#)
  - value, [1024](#)
- L4::lpc::Varg\_list< MAX >, [1025](#)
- L4::lpc::Varg\_list\_ref, [1026](#)
  - Varg\_list\_ref, [1027](#)
- L4::lpc\_gate, [1028](#)
  - get\_infos, [1031](#)
- L4::lpc\_svr, [666](#)
- L4::lpc\_svr::Br\_manager\_no\_buffers, [1031](#)
  - alloc\_buffer\_demand, [1034](#)
- L4::lpc\_svr::Compound\_reply, [1035](#)
- L4::lpc\_svr::Default\_loop\_hooks, [1036](#)
- L4::lpc\_svr::Default\_setup\_wait, [1037](#)
- L4::lpc\_svr::Default\_timeout, [1038](#)
- L4::lpc\_svr::Direct\_dispatch< R >, [1039](#)
- L4::lpc\_svr::Direct\_dispatch< R \* >, [1041](#)
- L4::lpc\_svr::Exc\_dispatch< R, Exc >, [1042](#)
- L4::lpc\_svr::Ignore\_errors, [1043](#)
- L4::lpc\_svr::Server\_iface, [1044](#)
  - add\_timeout, [1047](#)
  - alloc\_buffer\_demand, [1047](#)
  - get\_rcv\_cap, [1048](#)
  - rcv\_cap, [1049](#), [1050](#)
  - realloc\_rcv\_cap, [1051](#)
  - remove\_timeout, [1052](#)
- L4::lpc\_svr::Timed\_work< HOOKS >, [1053](#)
- L4::lpc\_svr::Timeout, [1053](#)
  - expired, [1055](#)
  - timeout, [1055](#)
- L4::lpc\_svr::Timeout\_queue, [1056](#)
  - add, [1057](#)
  - handle\_expired\_timeouts, [1057](#)
  - next\_timeout, [1058](#)
  - remove, [1058](#)
  - timeout\_expired, [1059](#)
- L4::lpc\_svr::Timeout\_queue\_hooks
  - add\_timeout, [1062](#)
  - remove\_timeout, [1062](#)
- L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_M↵AN >, [1060](#)
- L4::lrc, [1063](#)
  - attach, [1066](#)
  - detach, [1067](#)
  - receive, [1068](#)
  - unmask, [1069](#)
  - wait, [1070](#)
- L4::lrc\_eoi, [1070](#)
  - unmask, [1072](#)
- L4::lrc\_handler\_object, [1073](#)
- L4::lrc\_mux, [1076](#)
  - chain, [1078](#)
- L4::Kip::Mem\_desc, [1079](#)
  - all, [1082](#), [1083](#)
  - count, [1083](#), [1084](#)
  - end, [1084](#)
  - first, [1085](#)
  - Info\_sub\_type, [1081](#)
  - is\_virtual, [1086](#)
  - Mem\_desc, [1081](#)
  - Mem\_type, [1081](#)
  - set, [1086](#)
  - size, [1086](#)
  - start, [1087](#)
  - sub\_type, [1087](#)
  - type, [1088](#)
- L4::Kobject, [1088](#)
  - cap, [1089](#)
  - dec\_refcnt, [1091](#)
- L4::Kobject\_2t
  - \_\_iface, [1094](#)
  - \_\_iface\_list, [1094](#)
  - \_\_check\_protocols\_\_, [1095](#)
  - c, [1095](#)
  - Class, [1094](#)
- L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_↵DEMAND >, [1092](#)
- L4::Kobject\_3t
  - \_\_iface, [1098](#)
  - \_\_iface\_list, [1098](#)
  - \_\_check\_protocols\_\_, [1099](#)

- c, [1099](#)
- Class, [1098](#)
- L4::Kobject\_3t< Derived, Base1, Base2, Base3, PRO↵  
TO, S\_DEMAND >, [1096](#)
- L4::Kobject\_demand< T >, [1099](#)
- L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >, [1100](#)
- L4::Kobject\_typeid
  - Demand, [1103](#)
  - demand, [1103](#)
  - id, [1103](#)
  - proto\_dispatch, [1104](#)
- L4::Kobject\_typeid< T >, [1102](#)
- L4::Kobject\_typeid< void >, [1105](#)
- L4::Kobject\_x< Derived, ARGS >, [1106](#)
- L4::Meta, [1107](#)
  - interface, [1110](#)
  - num\_interfaces, [1110](#)
  - supports, [1110](#)
- L4::Out\_of\_memory, [1111](#)
- L4::Pager, [1114](#)
  - page\_fault, [1117](#)
- L4::Platform\_control, [1118](#)
  - cpu\_disable, [1120](#)
  - cpu\_enable, [1121](#)
  - Opcode, [1120](#)
  - system\_shutdown, [1121](#)
  - system\_suspend, [1121](#)
- L4::Poll\_timeout\_kipclock, [1122](#)
  - Poll\_timeout\_kipclock, [1123](#)
  - set, [1123](#)
  - test, [1124](#)
  - timed\_out, [1125](#)
- L4::Proto\_t< P >, [1125](#)
- L4::Rcv\_endpoint, [1127](#)
  - bind\_thread, [1130](#)
- L4::Registry\_iface, [1131](#)
  - register\_irq\_obj, [1132](#)
  - register\_obj, [1133](#), [1134](#)
  - unregister\_obj, [1135](#)
- L4::Runtime\_error, [1136](#)
  - err\_no, [1138](#)
  - extra\_str, [1138](#)
  - Runtime\_error, [1138](#)
- L4::Scheduler, [1139](#)
  - idle\_time, [1141](#)
  - info, [1142](#)
  - is\_online, [1142](#)
  - run\_thread, [1143](#)
- L4::Semaphore, [1144](#)
  - down, [1147](#)
  - up, [1148](#)
- L4::Server
  - internal\_loop, [1151](#)
  - loop, [1151](#)
  - Server, [1150](#)
- L4::Server< LOOP\_HOOKS >, [1149](#)
- L4::Server\_object, [1152](#)
  - dispatch, [1153](#)
- L4::Server\_object\_t
  - dispatch\_meta\_request, [1156](#)
  - get\_buffer\_demand, [1157](#)
  - proto\_dispatch, [1157](#)
- L4::Server\_object\_t< IFACE, BASE >, [1154](#)
- L4::Server\_object\_x< Derived, IFACE, BASE >, [1158](#)
- L4::Smart\_cap
  - Smart\_cap, [1163](#)
- L4::Smart\_cap< T, SMART >, [1160](#)
- L4::String, [1164](#)
- L4::Task, [1164](#)
  - add\_ku\_mem, [1167](#)
  - cap\_equal, [1168](#)
  - cap\_has\_child, [1168](#)
  - cap\_valid, [1169](#)
  - delete\_obj, [1169](#)
  - map, [1170](#)
  - release\_cap, [1170](#)
  - unmap, [1171](#)
  - unmap\_batch, [1172](#)
- L4::Thread, [1173](#)
  - control, [1176](#)
  - ex\_regs, [1176](#), [1177](#)
  - modify\_senders, [1178](#)
  - register\_del\_irq, [1179](#)
  - stats\_time, [1179](#)
  - switch\_to, [1180](#)
  - vcpu\_control, [1180](#)
  - vcpu\_control\_ext, [1181](#)
  - vcpu\_resume\_commit, [1182](#)
  - vcpu\_resume\_start, [1182](#)
- L4::Thread::Attr, [1183](#)
  - Attr, [1184](#)
  - bind, [1184](#)
  - exc\_handler, [1185](#)
  - pager, [1185](#), [1186](#)
  - ux\_host\_syscall, [1186](#)
- L4::Thread::Modify\_senders, [1186](#)
  - add, [1187](#)
- L4::Triggerable, [1188](#)
  - trigger, [1190](#)
- L4::Type\_info, [1192](#)
- L4::Type\_info::Demand, [1193](#)
  - Demand, [1194](#)
  - no\_demand, [1195](#)
- L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PO↵  
RTS >, [1196](#)
- L4::Type\_info::Demand\_union\_t< D1, D2 >, [1198](#)
- L4::Typeid, [667](#)
- L4::Typeid::Detail::\_Rpcs< OPCODE, O, Default\_op<  
R > >::Rpc< Y >, [1202](#)
- L4::Typeid::Detail::\_Rpcs< OPCODE, O, R, X... >, [1203](#)
- L4::Typeid::Detail::\_Rpcs< OPCODE, O, R, X... >::↵  
Rpc< Y >, [1204](#)
- L4::Typeid::Detail::\_Rpcs< OPCODE, O, X >, [1201](#)
- L4::Typeid::Detail::\_Rpcs\_end, [1205](#)

- L4::Typeid::P\_dispatch< LIST >, [1206](#)
- L4::Typeid::Raw\_ipc< CLASS >, [1207](#)
- L4::Typeid::Rpc\_nocode< OPERATION >, [1207](#)
- L4::Typeid::Rpcs< RPCS >, [1210](#)
- L4::Typeid::Rpcs\_code< OPCODE\_TYPE >, [1212](#)
- L4::Typeid::Rpcs\_code< OPCODE\_TYPE >::F< RP↵  
CS >, [1213](#)
- L4::Typeid::Rpcs\_sys< ARG >, [1215](#)
- L4::Types, [667](#)
- L4::Types::Bool< V >, [1217](#)
- L4::Types::False, [1218](#)
- L4::Types::Flags
  - clear, [1224](#)
  - Flags, [1223](#)
  - from\_raw, [1224](#)
  - None\_type, [1223](#)
- L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1220](#)
- L4::Types::Same< A, B >, [1225](#)
- L4::Types::True, [1227](#)
- L4::Unknown\_error, [1229](#)
- L4::Vcon, [1232](#)
  - get\_attr, [1234](#)
  - read, [1235](#)
  - read\_with\_flags, [1236](#)
  - send, [1236](#)
  - set\_attr, [1237](#)
  - write, [1238](#)
- L4::Vm, [1239](#)
- L4\_ALWAYS\_INLINE
  - Basic Macros, [107](#)
- L4\_DISABLE\_COPY
  - capability, [2070](#)
- L4\_HIDDEN
  - Basic Macros, [107](#)
- L4\_INLINE\_RPC\_NF\_OP
  - ipc\_iface, [2089](#)
- L4\_INLINE\_RPC\_NF
  - ipc\_iface, [2089](#)
- L4\_INLINE\_RPC\_OP
  - ipc\_iface, [2090](#)
- L4\_INLINE\_RPC
  - ipc\_iface, [2088](#)
- L4\_IPC\_TIMEOUT\_0
  - Timeouts, [551](#)
- L4\_LOG2\_PAGESIZE
  - Memory related, [422](#)
- L4\_LOG2\_SUPERPAGESIZE
  - Memory related, [422](#)
- L4\_NOTHROW
  - Basic Macros, [107](#)
- L4\_PAGEMASK
  - Memory related, [422](#)
- L4\_RPC\_DEF
  - ipc\_client, [2085](#)
- L4\_RPC\_NF\_OP
  - ipc\_iface, [2091](#)
- L4\_RPC\_NF
  - ipc\_iface, [2091](#)
- L4\_RPC\_OP
  - ipc\_iface, [2092](#)
- L4\_RPC
  - ipc\_iface, [2090](#)
- L4\_SUPERPAGEMASK
  - Memory related, [423](#)
- L4\_SUPERPAGESIZE
  - Memory related, [423](#)
- l4\_addr\_consts\_t
  - Memory related, [423](#)
- l4\_assert
  - sys/assert.h, [2253](#)
- l4\_buf\_regs\_t, [1242](#)
- l4\_buffer\_desc\_consts\_t
  - Buffer Registers (BRs), [117](#)
- l4\_busy\_wait\_ns
  - Timestamp Counter, [560](#)
- l4\_busy\_wait\_us
  - Timestamp Counter, [561](#)
- l4\_cache\_clean\_data
  - Cache Consistency, [124](#)
- l4\_cache\_coherent
  - Cache Consistency, [125](#)
- l4\_cache\_dma\_coherent
  - Cache Consistency, [125](#)
- l4\_cache\_flush\_data
  - Cache Consistency, [126](#)
- l4\_cache\_inv\_data
  - Cache Consistency, [126](#)
- l4\_calibrate\_tsc
  - Timestamp Counter, [561](#)
- l4\_cap\_consts\_t
  - Capabilities, [129](#)
- l4\_cap\_fpage\_rights
  - Flex pages, [236](#)
- l4\_capability\_equal
  - Capabilities, [130](#)
- l4\_capability\_next
  - l4/sys/types.h, [1822](#)
- l4\_debugger\_get\_object\_name
  - debugger.h, [2119](#)
- l4\_debugger\_global\_id
  - Kernel Debugger, [324](#)
- l4\_debugger\_kobj\_to\_id
  - Kernel Debugger, [325](#)
- l4\_debugger\_query\_log\_name
  - debugger.h, [2119](#)
- l4\_debugger\_query\_log\_typeid
  - debugger.h, [2120](#)
- l4\_debugger\_set\_object\_name
  - Kernel Debugger, [325](#)
- l4\_debugger\_switch\_log
  - debugger.h, [2120](#)
- l4\_default\_caps\_t
  - Capabilities, [129](#)
- l4\_error
  - Error Handling, [194](#)



- l4\_error\_code\_t
  - Error codes, [200](#)
- l4\_exc\_regs\_t, [1243](#)
  - flags, [1245](#)
  - ss, [1245](#)
- l4\_factory\_create\_factory
  - Factory, [213](#)
- l4\_factory\_create\_factory\_u
  - Factory, [214](#)
- l4\_factory\_create\_gate
  - Factory, [215](#)
- l4\_factory\_create\_gate\_u
  - Factory, [216](#)
- l4\_factory\_create\_irq
  - Factory, [217](#)
- l4\_factory\_create\_irq\_u
  - Factory, [218](#)
- l4\_factory\_create\_task
  - Factory, [219](#)
- l4\_factory\_create\_task\_u
  - Factory, [220](#)
- l4\_factory\_create\_thread
  - Factory, [221](#)
- l4\_factory\_create\_thread\_u
  - Factory, [222](#)
- l4\_factory\_create\_vm
  - Factory, [223](#)
- l4\_factory\_create\_vm\_u
  - Factory, [224](#)
- l4\_fpage
  - Flex pages, [240](#)
- l4\_fpage\_all
  - Flex pages, [240](#)
- l4\_fpage\_cacheability\_opt\_t
  - Flex pages, [238](#)
- l4\_fpage\_consts
  - Flex pages, [238](#)
- l4\_fpage\_contains
  - Flex pages, [241](#)
- l4\_fpage\_invalid
  - Flex pages, [241](#)
- l4\_fpage\_ioport
  - Flex pages, [242](#)
- l4\_fpage\_max\_order
  - Flex pages, [242](#)
- l4\_fpage\_memaddr
  - Flex pages, [243](#)
- l4\_fpage\_obj
  - Flex pages, [244](#)
- l4\_fpage\_page
  - Flex pages, [245](#)
- L4\_fpage\_rights
  - Flex pages, [239](#)
- l4\_fpage\_rights
  - Flex pages, [245](#)
- l4\_fpage\_set\_rights
  - Flex pages, [246](#)
- l4\_fpage\_size
  - Flex pages, [246](#)
- l4\_fpage\_t, [1246](#)
- l4\_fpage\_type
  - Flex pages, [247](#)
- l4\_get\_hz
  - Timestamp Counter, [562](#)
- l4\_icu\_bind
  - Interrupt controller, [312](#)
- l4\_icu\_bind\_u
  - Interrupt controller, [313](#)
- L4\_icu\_flags
  - Interrupt controller, [312](#)
- l4\_icu\_info
  - Interrupt controller, [314](#)
- l4\_icu\_info\_t, [1246](#)
  - features, [1248](#)
  - Interrupt controller, [311](#)
- l4\_icu\_info\_u
  - Interrupt controller, [314](#)
- l4\_icu\_mask
  - Interrupt controller, [315](#)
- l4\_icu\_mask\_u
  - Interrupt controller, [316](#)
- l4\_icu\_msi\_info
  - Interrupt controller, [317](#)
- l4\_icu\_msi\_info\_t, [1248](#)
  - msi\_data, [1249](#)
- l4\_icu\_msi\_info\_u
  - Interrupt controller, [318](#)
- L4\_icu\_opcode
  - L4 IPC Opcodes, [340](#)
- l4\_icu\_set\_mode
  - Interrupt controller, [318](#)
- l4\_icu\_set\_mode\_u
  - Interrupt controller, [319](#)
- l4\_icu\_unbind
  - Interrupt controller, [320](#)
- l4\_icu\_unbind\_u
  - Interrupt controller, [321](#)
- l4\_icu\_unmask
  - Interrupt controller, [322](#)
- l4\_icu\_unmask\_u
  - Interrupt controller, [322](#)
- l4\_int16\_t
  - Integer Types, [291](#)
- l4\_int32\_t
  - Integer Types, [291](#)
- l4\_int64\_t
  - Integer Types, [292](#)
- l4\_int8\_t
  - Integer Types, [292](#)
- l4\_iofpage
  - Flex pages, [247](#)
- l4\_ipc
  - Object Invocation, [456](#)
- l4\_ipc\_call
  - Object Invocation, [457](#)
- l4\_ipc\_error



- Error Handling, [195](#)
- `l4_ipc_error_code`
  - Error Handling, [197](#)
- `l4_ipc_gate_bind_thread`
  - IPC-Gate API, [267](#)
- `l4_ipc_gate_get_infos`
  - IPC-Gate API, [267](#)
- `L4_ipc_gate_ops`
  - L4 IPC Opcodes, [341](#)
- `l4_ipc_is_rcv_error`
  - Error Handling, [198](#)
- `l4_ipc_is_snd_error`
  - Error Handling, [198](#)
- `l4_ipc_receive`
  - Object Invocation, [459](#)
- `l4_ipc_reply_and_wait`
  - Object Invocation, [460](#)
- `l4_ipc_send`
  - Object Invocation, [461](#)
- `l4_ipc_send_and_wait`
  - Object Invocation, [462](#)
- `l4_ipc_sleep`
  - Object Invocation, [464](#)
- `l4_ipc_tcr_error_t`
  - Error Handling, [194](#)
- `l4_ipc_timeout`
  - Timeouts, [552](#)
- `l4_ipc_to_errno`
  - `l4/sys/ipc.h`, [2147](#)
- `l4_ipc_wait`
  - Object Invocation, [464](#)
- `l4_irq_attach`
  - IRQs, [272](#)
- `l4_irq_attach_u`
  - IRQs, [272](#)
- `l4_irq_detach`
  - IRQs, [273](#)
- `l4_irq_detach_u`
  - IRQs, [274](#)
- `L4_irq_mode`
  - IRQs, [271](#)
- `l4_irq_mux_chain`
  - IRQs, [275](#)
- `l4_irq_mux_chain_u`
  - IRQs, [276](#)
- `l4_irq_receive`
  - IRQs, [277](#)
- `l4_irq_receive_u`
  - IRQs, [278](#)
- `l4_irq_trigger`
  - IRQs, [279](#)
- `l4_irq_trigger_u`
  - IRQs, [280](#)
- `l4_irq_unmask`
  - IRQs, [281](#)
- `l4_irq_unmask_u`
  - IRQs, [281](#)
- `l4_irq_wait`
  - IRQs, [282](#)
- `l4_irq_wait_u`
  - IRQs, [283](#)
- `l4_is_fpage_writable`
  - Flex pages, [248](#)
- `l4_is_invalid_cap`
  - Capabilities, [130](#)
- `l4_is_valid_cap`
  - Capabilities, [131](#)
- `l4_kernel_info_get_mem_desc_end`
  - Memory descriptors (C version), [415](#)
- `l4_kernel_info_get_mem_desc_is_virtual`
  - Memory descriptors (C version), [415](#)
- `l4_kernel_info_get_mem_desc_start`
  - Memory descriptors (C version), [415](#)
- `l4_kernel_info_get_mem_desc_subtype`
  - Memory descriptors (C version), [416](#)
- `l4_kernel_info_get_mem_desc_type`
  - Memory descriptors (C version), [416](#)
- `l4_kernel_info_get_num_mem_descs`
  - Memory descriptors (C version), [416](#)
- `l4_kernel_info_mem_desc_t`, [1249](#)
  - Memory descriptors (C version), [414](#)
- `l4_kernel_info_set_mem_desc`
  - Memory descriptors (C version), [417](#)
- `l4_kernel_info_t`, [1250](#)
- `l4_kernel_info_version_offset`
  - Kernel Interface Page, [328](#)
- `l4_kip_clock`
  - Kernel Interface Page, [329](#)
- `l4_kip_clock_lw`
  - Kernel Interface Page, [330](#)
- `l4_kip_version`
  - Kernel Interface Page, [330](#)
- `l4_kip_version_string`
  - Kernel Interface Page, [331](#)
- `l4_map_control`
  - Message Items, [429](#)
- `l4_map_obj_control`
  - Message Items, [430](#)
- `l4_mem_info_sub_type_t`
  - Memory descriptors (C version), [414](#)
- `L4_mem_op_widths`
  - Memory operations., [418](#)
- `l4_mem_read`
  - Memory operations., [419](#)
- `l4_mem_type_t`
  - Memory descriptors (C version), [415](#)
- `l4_mem_write`
  - Memory operations., [419](#)
- `l4_msg_item_consts_t`
  - Message Items, [428](#)
- `l4_msg_regs_t`, [1252](#)
- `l4_msgtag`
  - Message Tag, [435](#)
- `l4_msgtag_flags`
  - Message Tag, [434](#), [437](#)
- `l4_msgtag_has_error`

- Message Tag, [438](#)
- `l4_msgtag_is_exception`
  - Message Tag, [439](#)
- `l4_msgtag_is_io_page_fault`
  - Message Tag, [440](#)
- `l4_msgtag_is_page_fault`
  - Message Tag, [441](#)
- `l4_msgtag_is_preemption`
  - Message Tag, [442](#)
- `l4_msgtag_is_sigma0`
  - Message Tag, [443](#)
- `l4_msgtag_is_sys_exception`
  - Message Tag, [444](#)
- `l4_msgtag_items`
  - Message Tag, [445](#)
- `l4_msgtag_label`
  - Message Tag, [446](#)
- `l4_msgtag_protocol`
  - Message Tag, [434](#)
- `l4_msgtag_t`, [1253](#)
  - flags, [1255](#)
  - Message Tag, [433](#)
- `l4_msgtag_words`
  - Message Tag, [447](#)
- `l4_ns_to_tsc`
  - Timestamp Counter, [562](#)
- `l4_obj_fpage`
  - Flex pages, [248](#)
- `L4_obj_fpage_ctl`
  - Flex pages, [239](#)
- `l4_platform_ctl_cpu_disable`
  - Platform Control C API, [468](#)
- `l4_platform_ctl_cpu_enable`
  - Platform Control C API, [469](#)
- `L4_platform_ctl_ops`
  - L4 IPC Opcodes, [342](#)
- `L4_platform_ctl_proto`
  - Message Tag, [435](#)
- `l4_platform_ctl_system_shutdown`
  - Platform Control C API, [469](#)
- `l4_platform_ctl_system_suspend`
  - Platform Control C API, [470](#)
- `l4_rcv_ep_bind_thread`
  - IPC-Gate API, [268](#)
- `L4_rcv_ep_ops`
  - `rcv_endpoint.h`, [2192](#)
- `l4_rcv_timeout`
  - Timeouts, [553](#)
- `l4_rdpmc`
  - Timestamp Counter, [563](#)
- `l4_rdpmc_32`
  - Timestamp Counter, [563](#)
- `l4_rdtsc`
  - Timestamp Counter, [564](#)
- `l4_rdtsc_32`
  - Timestamp Counter, [564](#)
- `l4_round_page`
  - Memory related, [424](#)
- `l4_round_size`
  - Memory related, [425](#)
- `l4_sched_cpu_set`
  - Scheduler, [492](#)
- `l4_sched_cpu_set_t`, [1255](#)
  - `gran_offset`, [1257](#)
  - granularity, [1256](#)
  - offset, [1256](#)
- `l4_sched_param_t`, [1258](#)
- `l4_scheduler_idle_time`
  - Scheduler, [493](#)
- `l4_scheduler_info`
  - Scheduler, [494](#)
- `l4_scheduler_is_online`
  - Scheduler, [495](#)
- `L4_scheduler_ops`
  - Scheduler, [492](#)
- `l4_scheduler_run_thread`
  - Scheduler, [495](#)
- `l4_sleep`
  - `amd64/l4/util/util.h`, [1754](#)
  - Utility Functions, [569](#)
  - `x86/l4/util/util.h`, [1757](#)
- `l4_snd_fpage_t`, [1259](#)
- `l4_snd_timeout`
  - Timeouts, [553](#)
- `l4_sndfpage_add`
  - Object Invocation, [466](#)
- `l4_syscall_flags_t`
  - Object Invocation, [456](#)
- `l4_task_add_ku_mem`
  - Task, [521](#)
- `l4_task_cap_equal`
  - Task, [521](#)
- `l4_task_cap_has_child`
  - Task, [521](#)
- `l4_task_cap_valid`
  - Task, [522](#)
- `l4_task_delete_obj`
  - Task, [522](#)
- `L4_task_ldt_x86_consts`
  - `amd64/l4/sys/segment.h`, [1763](#)
  - `x86/l4/sys/segment.h`, [1769](#)
- `l4_task_map`
  - Task, [523](#)
- `L4_task_ops`
  - L4 IPC Opcodes, [342](#)
- `l4_task_release_cap`
  - Task, [524](#)
- `l4_task_unmap`
  - Task, [524](#)
- `l4_task_unmap_batch`
  - Task, [525](#)
- `l4_thread_arm_set_tpidruro`
  - Thread, [531](#)
- `l4_thread_control_alien`
  - Thread control, [546](#)
- `l4_thread_control_bind`

- Thread control, [546](#)
- `l4_thread_control_commit`
  - Thread control, [547](#)
- `l4_thread_control_exc_handler`
  - Thread control, [547](#)
- `L4_thread_control_flags`
  - Thread, [529](#)
- `L4_thread_control_mr_indices`
  - Thread, [529](#)
- `l4_thread_control_pager`
  - Thread control, [548](#)
- `l4_thread_control_start`
  - Thread control, [548](#)
- `l4_thread_control_ux_host_syscall`
  - Thread control, [548](#)
- `l4_thread_ex_regs`
  - Thread, [532](#)
- `L4_thread_ex_regs_flags`
  - Thread, [531](#)
- `l4_thread_ex_regs_ret`
  - Thread, [533](#)
- `l4_thread_ex_regs_ret_u`
  - Thread, [534](#)
- `l4_thread_ex_regs_u`
  - Thread, [535](#)
- `l4_thread_longjmp`
  - `amd64/l4f/l4/util/setjmp.h`, [1780](#)
  - `x86/l4f/l4/util/setjmp.h`, [1783](#)
- `l4_thread_modify_sender_add`
  - Thread, [536](#)
- `l4_thread_modify_sender_commit`
  - Thread, [536](#)
- `l4_thread_modify_sender_start`
  - Thread, [537](#)
- `L4_thread_ops`
  - L4 IPC Opcodes, [342](#)
- `l4_thread_register_del_irq`
  - Thread, [537](#)
- `l4_thread_regs_t`, [1260](#)
- `l4_thread_setjmp`
  - `amd64/l4f/l4/util/setjmp.h`, [1781](#)
  - `x86/l4f/l4/util/setjmp.h`, [1784](#)
- `l4_thread_stats_time`
  - Thread, [538](#)
- `l4_thread_switch`
  - Thread, [538](#)
- `l4_thread_vcpu_control`
  - Thread, [538](#)
- `l4_thread_vcpu_control_ext`
  - Thread, [539](#)
- `l4_thread_vcpu_control_ext_u`
  - Thread, [540](#)
- `l4_thread_vcpu_control_u`
  - Thread, [541](#)
- `l4_thread_vcpu_resume_commit`
  - Thread, [542](#)
- `l4_thread_vcpu_resume_start`
  - Thread, [543](#)
- `l4_thread_yield`
  - Thread, [543](#)
- `l4_timeout`
  - Timeouts, [554](#)
- `l4_timeout_abs`
  - Timeouts, [554](#)
- `l4_timeout_abs_validity`
  - Timeouts, [552](#)
- `l4_timeout_get`
  - Timeouts, [555](#)
- `l4_timeout_is_absolute`
  - Timeouts, [556](#)
- `l4_timeout_rel`
  - Timeouts, [556](#)
- `l4_timeout_rel_get`
  - Timeouts, [557](#)
- `l4_timeout_s`, [1261](#)
  - Timeouts, [552](#)
- `l4_timeout_t`, [1262](#)
  - Timeouts, [552](#)
- `l4_touch_ro`
  - Utility Functions, [570](#)
- `l4_touch_rw`
  - Utility Functions, [570](#)
- `l4_tracebuffer_status_t`, [1263](#)
  - `cnt_jobmap_tlb_flush`, [1264](#)
- `l4_tracebuffer_status_window_t`, [1265](#)
- `l4_trunc_page`
  - Memory related, [426](#)
- `l4_trunc_size`
  - Memory related, [426](#)
- `l4_tsc_init`
  - Timestamp Counter, [565](#)
- `l4_tsc_to_ns`
  - Timestamp Counter, [566](#)
- `l4_tsc_to_s_and_ns`
  - Timestamp Counter, [566](#)
- `l4_tsc_to_us`
  - Timestamp Counter, [567](#)
- `l4_uint16_t`
  - Integer Types, [292](#)
- `l4_uint32_t`
  - Integer Types, [292](#)
- `l4_uint64_t`
  - Integer Types, [292](#)
- `l4_uint8_t`
  - Integer Types, [292](#)
- `l4_unmap_flags_t`
  - Task, [520](#)
- `l4_usleep`
  - Utility Functions, [571](#)
- `l4_utcb_br`
  - Virtual Registers (UTCBs), [620](#)
- `L4_utcb_consts_x86`
  - x86 Virtual Registers (UTCB), [634](#)
- `l4_utcb_exc`
  - Exception registers, [207](#)
- `l4_utcb_exc_is_ex_regs_exception`

- Exception registers, [207](#)
- `l4_utcb_exc_is_pf`
  - Exception registers, [208](#)
- `l4_utcb_exc_pc`
  - Exception registers, [208](#)
- `l4_utcb_exc_pc_set`
  - Exception registers, [209](#)
- `l4_utcb_mr`
  - Virtual Registers (UTCBS), [620](#)
- `l4_utcb_mr64_idx`
  - Timeouts, [557](#)
- `l4_utcb_t`
  - Virtual Registers (UTCBS), [619](#)
- `l4_utcb_tcr`
  - Virtual Registers (UTCBS), [621](#)
- `l4_vcon_attr_t`, [1266](#)
- `l4_vcon_get_attr`
  - Virtual Console, [607](#)
- `l4_vcon_get_attr_u`
  - Virtual Console, [607](#)
- `L4_vcon_i_flags`
  - Virtual Console, [605](#)
- `L4_vcon_l_flags`
  - Virtual Console, [606](#)
- `L4_vcon_o_flags`
  - Virtual Console, [606](#)
- `L4_vcon_ops`
  - L4 IPC Opcodes, [343](#)
- `l4_vcon_read`
  - Virtual Console, [608](#)
- `L4_vcon_read_flags`
  - `vcon.h`, [2239](#)
- `l4_vcon_read_u`
  - Virtual Console, [609](#)
- `l4_vcon_read_with_flags`
  - Virtual Console, [610](#)
- `l4_vcon_send`
  - Virtual Console, [611](#)
- `l4_vcon_send_u`
  - Virtual Console, [612](#)
- `l4_vcon_set_attr`
  - Virtual Console, [613](#)
- `l4_vcon_set_attr_u`
  - Virtual Console, [613](#)
- `L4_vcon_size_consts`
  - Virtual Console, [606](#)
- `l4_vcon_write`
  - Virtual Console, [614](#)
- `l4_vcon_write_u`
  - Virtual Console, [615](#)
- `l4_vcpu_ipc_regs_t`, [1267](#)
- `l4_vcpu_regs_t`, [1268](#)
  - `ax`, [1269](#)
  - `bp`, [1270](#)
  - `bx`, [1270](#)
  - `cx`, [1270](#)
  - `di`, [1270](#)
  - `dx`, [1271](#)
  - `si`, [1271](#)
- `L4_vcpu_state_flags`
  - vCPU API, [624](#)
- `L4_vcpu_state_offset`
  - vCPU API, [624](#)
- `l4_vcpu_state_t`, [1272](#)
- `L4_vcpu_sticky_flags`
  - vCPU API, [625](#)
- `l4_vhw_descriptor`, [1274](#)
  - `count`, [1276](#)
  - `descs`, [1276](#)
  - `magic`, [1276](#)
  - `version`, [1276](#)
- `l4_vhw_entry`, [1277](#)
  - `fd`, [1278](#)
  - `irq_no`, [1278](#)
  - `mem_size`, [1278](#)
  - `mem_start`, [1278](#)
  - `provider_pid`, [1279](#)
  - `type`, [1279](#)
- `l4_vhw_entry_type`
  - Fiasco-UX Virtual devices, [232](#)
- `L4_virtio_cmd`
  - L4 VIRTIO Transport Layer, [349](#)
- `L4_virtio_irq_status`
  - L4 VIRTIO Transport Layer, [349](#)
- `L4_virtio_opcodes`
  - L4 VIRTIO Transport Layer, [350](#)
- `l4_vm_svm_vmcb_control_area`, [1280](#)
- `l4_vm_svm_vmcb_state_save_area`, [1280](#)
- `l4_vm_svm_vmcb_state_save_area_seg`, [1282](#)
- `l4_vm_svm_vmcb_t`, [1282](#)
- `l4_vm_tz_state`, [1284](#)
- `L4_vm_vmx_caps_regs`
  - VM API for VMX, [579](#)
- `l4_vm_vmx_clear`
  - VM API for VMX, [580](#)
- `L4_vm_vmx_dfl1_regs`
  - VM API for VMX, [579](#)
- `l4_vm_vmx_field_len`
  - VM API for VMX, [580](#)
- `l4_vm_vmx_field_order`
  - VM API for VMX, [581](#)
- `l4_vm_vmx_get_caps`
  - VM API for VMX, [582](#)
- `l4_vm_vmx_get_caps_default1`
  - VM API for VMX, [582](#)
- `l4_vm_vmx_get_cr2_index`
  - VM API for VMX, [583](#)
- `l4_vm_vmx_ptr_load`
  - VM API for VMX, [583](#)
- `l4_vm_vmx_read`
  - VM API for VMX, [584](#)
- `l4_vm_vmx_read_16`
  - VM API for VMX, [585](#)
- `l4_vm_vmx_read_32`
  - VM API for VMX, [586](#)
- `l4_vm_vmx_read_64`

- VM API for VMX, [586](#)
- `l4_vm_vmx_read_nat`
  - VM API for VMX, [587](#)
- `l4_vm_vmx_write`
  - VM API for VMX, [587](#)
- `l4_vm_vmx_write_16`
  - VM API for VMX, [588](#)
- `l4_vm_vmx_write_32`
  - VM API for VMX, [589](#)
- `l4_vm_vmx_write_64`
  - VM API for VMX, [590](#)
- `l4_vm_vmx_write_nat`
  - VM API for VMX, [590](#)
- `L4RE_ELF_AUX_ELEM_T`
  - L4Re ELF Auxiliary Information, [378](#)
- `L4RE_ELF_AUX_ELEM`
  - L4Re ELF Auxiliary Information, [378](#)
- `L4Re`, [668](#)
  - `chkcapi`, [674](#)
  - `chkipc`, [674](#)
  - `chksys`, [676–678](#)
  - `make_shared_cap`, [679](#)
  - `make_shared_del_cap`, [680](#)
  - `make_unique_cap`, [680](#)
  - `make_unique_del_cap`, [681](#)
  - `Shared_cap`, [670](#)
  - `shared_cap`, [670](#)
  - `Shared_del_cap`, [671](#)
  - `shared_del_cap`, [671](#)
  - `Unique_cap`, [672](#)
  - `unique_cap`, [672](#)
  - `Unique_del_cap`, [673](#)
  - `unique_del_cap`, [673](#)
- `L4Re C Interface`, [369](#)
  - `l4re_inhibitor_acquire`, [370](#)
- `L4Re C++ Interface`, [372](#)
- `L4Re Capability API`, [374](#)
  - `cap_alloc`, [376](#)
  - `make_auto_cap`, [375](#)
  - `make_auto_del_cap`, [375](#)
  - `make_ref_cap`, [375](#)
  - `make_ref_del_cap`, [376](#)
- `L4Re ELF Auxiliary Information`, [377](#)
  - `L4RE_ELF_AUX_ELEM_T`, [378](#)
  - `L4RE_ELF_AUX_ELEM`, [378](#)
- `L4Re Protocol identifiers`, [380](#)
- `L4Re Util C Interface`, [381](#)
- `L4Re Util C++ Interface`, [382](#)
- `L4Re::Cap_alloc`, [1284](#)
  - `alloc`, [1285](#), [1286](#)
  - `free`, [1287](#)
  - `get_cap_alloc`, [1288](#)
- `L4Re::Console`, [1289](#)
- `L4Re::Dataspace`, [1291](#)
  - `allocate`, [1294](#)
  - `clear`, [1294](#)
  - `copy_in`, [1295](#)
  - `flags`, [1295](#)
  - `info`, [1297](#)
  - `map`, [1297](#)
  - `Map_flags`, [1293](#)
  - `map_region`, [1298](#)
  - `phys`, [1299](#)
  - `size`, [1300](#)
- `L4Re::Dataspace::Stats`, [1300](#)
- `L4Re::Debug_obj`, [1301](#)
  - `debug`, [1303](#)
- `L4Re::Dma_space`, [1303](#)
  - `associate`, [1306](#)
  - `Attribute`, [1305](#)
  - `Attributes`, [1304](#)
  - `Direction`, [1305](#)
  - `disassociate`, [1306](#)
  - `map`, [1307](#)
  - `Space_attrib`, [1306](#)
  - `unmap`, [1307](#)
- `L4Re::Env`, [1308](#)
  - `env`, [1311](#)
  - `factory`, [1311](#), [1312](#)
  - `first_free_cap`, [1312](#)
  - `first_free_utcb`, [1313](#)
  - `get`, [1313](#)
  - `get_cap`, [1314](#), [1315](#)
  - `initial_caps`, [1315](#)
  - `log`, [1316](#)
  - `main_thread`, [1316](#), [1317](#)
  - `mem_alloc`, [1317](#)
  - `parent`, [1318](#)
  - `rm`, [1318](#), [1319](#)
  - `scheduler`, [1319](#)
  - `task`, [1320](#)
  - `utcb_area`, [1320](#)
- `L4Re::Event`, [1321](#)
  - `get_buffer`, [1324](#)
- `L4Re::Event_buffer_t`
  - `Event_buffer_t`, [1326](#)
  - `next`, [1327](#)
  - `put`, [1327](#)
- `L4Re::Event_buffer_t< PAYLOAD >`, [1325](#)
- `L4Re::Event_buffer_t< PAYLOAD >::Event`, [1328](#)
- `L4Re::Inhibitor`, [1329](#)
  - `acquire`, [1332](#)
  - `next_lock_info`, [1333](#)
  - `release`, [1334](#)
- `L4Re::Log`, [1334](#)
  - `print`, [1337](#)
  - `println`, [1337](#)
- `L4Re::Mem_alloc`, [1337](#)
  - `alloc`, [1341](#)
  - `free`, [1341](#)
  - `Mem_alloc_flags`, [1340](#)
- `L4Re::Mmio_space`, [1342](#)
  - `Access_width`, [1345](#)
  - `mmio_read`, [1345](#)
  - `mmio_write`, [1346](#)
- `L4Re::Namespace`, [1347](#)

- query, [1350](#), [1351](#)
- Query\_result\_flags, [1349](#)
- Register\_flags, [1349](#)
- register\_obj, [1352](#)
- unlink, [1352](#)
- L4Re::Ned::Cmd\_control, [1353](#)
  - execute, [1354](#)
- L4Re::Parent, [1355](#)
  - signal, [1357](#)
- L4Re::Rm, [1357](#)
  - attach, [1362](#), [1363](#)
  - Attach\_flags, [1361](#)
  - detach, [1365](#), [1366](#)
  - Detach\_flags, [1361](#)
  - Detach\_result, [1361](#)
  - find, [1367](#)
  - free\_area, [1368](#)
  - Region\_flags, [1362](#)
  - reserve\_area, [1368](#), [1369](#)
- L4Re::Smart\_cap\_auto< Unmap\_flags >, [1370](#)
- L4Re::Smart\_count\_cap< Unmap\_flags >, [1371](#)
- L4Re::Util, [682](#)
  - make\_shared\_cap, [690](#)
  - make\_shared\_del\_cap, [690](#)
  - make\_unique\_cap, [691](#)
  - make\_unique\_del\_cap, [691](#)
  - Shared\_cap, [684](#)
  - shared\_cap, [685](#)
  - Shared\_del\_cap, [686](#)
  - shared\_del\_cap, [686](#)
  - Unique\_cap, [687](#)
  - unique\_cap, [688](#)
  - Unique\_del\_cap, [688](#)
  - unique\_del\_cap, [689](#)
- L4Re::Util::Auto\_cap< T >, [1372](#)
- L4Re::Util::Auto\_del\_cap< T >, [1373](#)
- L4Re::Util::Br\_manager, [1374](#)
  - alloc\_buffer\_demand, [1377](#)
  - get\_rcv\_cap, [1377](#)
  - realloc\_rcv\_cap, [1378](#)
  - set\_rcv\_cap\_flags, [1378](#)
- L4Re::Util::Br\_manager\_hooks, [1379](#)
- L4Re::Util::Br\_manager\_timeout\_hooks, [1380](#)
- L4Re::Util::Cap\_alloc\_base, [1383](#)
- L4Re::Util::Counter< COUNTER >, [1384](#)
- L4Re::Util::Counting\_cap\_alloc
  - alloc, [1386](#), [1387](#)
  - Counting\_cap\_alloc, [1386](#)
  - free, [1388](#)
  - release, [1389](#)
  - setup, [1390](#)
  - take, [1391](#)
- L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE >, [1384](#)
- L4Re::Util::Dataspace\_svr, [1391](#)
  - allocate, [1393](#)
  - clear, [1394](#)
  - copy, [1394](#)
  - is\_static, [1395](#)
  - map, [1396](#)
  - map\_hook, [1397](#)
  - page\_shift, [1398](#)
  - phys, [1399](#)
  - release, [1400](#)
  - take, [1400](#)
- L4Re::Util::Event\_buffer\_consumer\_t
  - foreach\_available\_event, [1404](#)
  - process, [1404](#)
- L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, [1401](#)
- L4Re::Util::Event\_buffer\_t
  - attach, [1407](#)
  - buf, [1408](#)
  - detach, [1408](#)
- L4Re::Util::Event\_buffer\_t< PAYLOAD >, [1405](#)
- L4Re::Util::Event\_svr< SVR >, [1409](#)
- L4Re::Util::Event\_t
  - buffer, [1412](#)
  - init, [1413](#)
  - init\_poll, [1414](#)
  - irq, [1415](#)
  - Mode, [1412](#)
- L4Re::Util::Event\_t< PAYLOAD >, [1411](#)
- L4Re::Util::Item\_alloc\_base, [1415](#)
- L4Re::Util::Names::Name, [1416](#)
- L4Re::Util::Names::Name\_space, [1417](#)
  - alloc\_dynamic\_entry, [1419](#)
  - copy\_receive\_cap, [1419](#)
  - free\_capability, [1419](#)
  - free\_dynamic\_entry, [1420](#)
  - free\_epiface, [1420](#)
  - get\_epiface, [1420](#)
- L4Re::Util::Object\_registry, [1421](#)
  - Object\_registry, [1423](#)
  - register\_irq\_obj, [1424](#)
  - register\_obj, [1425](#), [1426](#)
  - unregister\_obj, [1426](#)
- L4Re::Util::Ref\_cap< T >, [1427](#)
- L4Re::Util::Ref\_del\_cap< T >, [1428](#)
- L4Re::Util::Registry\_server
  - registry, [1432](#), [1433](#)
  - Registry\_server, [1432](#)
- L4Re::Util::Registry\_server< LOOP\_HOOKS >, [1429](#)
- L4Re::Util::Smart\_cap\_auto< Unmap\_flags >, [1433](#)
- L4Re::Util::Smart\_count\_cap< Unmap\_flags >, [1434](#)
- L4Re::Util::Vcon\_svr< SVR >, [1435](#)
- L4Re::Util::Video::Goos\_svr, [1436](#)
  - get\_fb, [1437](#)
  - init\_infos, [1437](#)
  - refresh, [1438](#)
  - screen\_info, [1439](#)
  - view\_info, [1439](#)
- L4Re::Vfs, [692](#)
- L4Re::Vfs::Be\_file, [1440](#)
  - data\_space, [1443](#)
  - fstat64, [1443](#)

- unlock\_all\_locks, 1443
- L4Re::Vfs::Be\_file\_system, 1444
  - ~Be\_file\_system, 1446
  - Be\_file\_system, 1446
  - type, 1446
- L4Re::Vfs::Directory, 1447
  - faccessat, 1449
  - link, 1449
  - mkdir, 1449
  - rename, 1450
  - rmdir, 1450
  - symlink, 1451
  - unlink, 1451
- L4Re::Vfs::File, 1452
- L4Re::Vfs::File\_system, 1454
  - mount, 1455
  - type, 1456
- L4Re::Vfs::Fs, 1456
  - alloc\_fd, 1459
  - free\_fd, 1459
  - get\_file, 1459
  - mount, 1460
  - set\_fd, 1460
- L4Re::Vfs::Generic\_file, 1461
  - fchmod, 1463
  - fstat64, 1464
  - get\_status\_flags, 1464
  - set\_status\_flags, 1464
  - unlock\_all\_locks, 1465
- L4Re::Vfs::Mman, 1466
- L4Re::Vfs::Ops, 1467
- L4Re::Vfs::Regular\_file, 1470
  - data\_space, 1472
  - fdatsync, 1472
  - fsync, 1472
  - ftruncate64, 1472
  - get\_lock, 1473
  - lseek64, 1473
  - readv, 1473
  - set\_lock, 1474
  - writev, 1474
- L4Re::Vfs::Special\_file, 1475
  - ioctl, 1477
- L4Re::Video::Color\_component, 1478
  - Color\_component, 1479
  - dump, 1479
  - get, 1479
  - operator==, 1480
  - set, 1480
  - shift, 1480
  - size, 1481
- L4Re::Video::Goos, 1482
  - create\_buffer, 1485
  - create\_view, 1485
  - delete\_buffer, 1485
  - delete\_view, 1486
  - Flags, 1484
  - get\_static\_buffer, 1486
  - info, 1487
  - view, 1487
- L4Re::Video::Goos::Info, 1488
  - auto\_refresh, 1489
- L4Re::Video::Pixel\_info, 1490
  - a, 1492
  - b, 1493
  - bits\_per\_pixel, 1493
  - bytes\_per\_pixel, 1494
  - dump, 1494
  - g, 1495
  - has\_alpha, 1496
  - operator==, 1496
  - Pixel\_info, 1491, 1492
  - r, 1497
- L4Re::Video::View, 1497
  - Flags, 1499
  - info, 1500
  - refresh, 1500
  - set\_info, 1501
  - set\_viewport, 1501
  - stack, 1502
  - V\_flags, 1499
- L4Re::Video::View::Info, 1502
- L4UTIL\_MB\_MEMORY
  - mb\_info.h, 2310
- l4io\_device\_types\_t
  - IO interface, 259
- l4io\_get\_root\_device
  - io.h, 1817
- l4io\_has\_resource
  - IO interface, 260
- l4io\_iomem\_flags\_t
  - IO interface, 259
- l4io\_iterate\_devices
  - io.h, 1817
- l4io\_lookup\_device
  - IO interface, 261
- l4io\_lookup\_resource
  - IO interface, 261
- l4io\_release\_iomem
  - IO interface, 262
- l4io\_release\_ioport
  - IO interface, 262
- l4io\_request\_all\_ioports
  - io.h, 1818
- l4io\_request\_icu
  - io.h, 1818
- l4io\_request\_iomem
  - IO interface, 262
- l4io\_request\_iomem\_region
  - IO interface, 263
- l4io\_request\_ioport
  - IO interface, 264
- l4io\_request\_resource\_iomem
  - IO interface, 264
- l4io\_resource\_t
  - IO interface, 259

- l4io\_resource\_types\_t
  - IO interface, [260](#)
- l4io\_search\_iomem\_region
  - IO interface, [265](#)
- l4irq\_attach
  - Interface using direct functionality., [297](#)
- l4irq\_attach\_cap
  - Interface using direct functionality., [303](#)
- l4irq\_attach\_cap\_ft
  - Interface using direct functionality., [303](#)
- l4irq\_attach\_ft
  - Interface using direct functionality., [298](#)
- l4irq\_attach\_thread
  - Interface using direct functionality., [298](#)
- l4irq\_attach\_thread\_cap
  - Interface using direct functionality., [304](#)
- l4irq\_attach\_thread\_cap\_ft
  - Interface using direct functionality., [304](#)
- l4irq\_attach\_thread\_ft
  - Interface using direct functionality., [299](#)
- l4irq\_detach
  - Interface using direct functionality., [299](#)
- l4irq\_release
  - Interface for asynchronous ISR handlers., [295](#)
- l4irq\_request
  - Interface for asynchronous ISR handlers., [296](#)
- l4irq\_request\_cap
  - Interface for asynchronous ISR handlers with a given IRQ capability., [293](#)
- l4irq\_unmask
  - Interface using direct functionality., [300](#)
- l4irq\_unmask\_and\_wait\_any
  - Interface using direct functionality., [300](#)
- l4irq\_wait
  - Interface using direct functionality., [300](#)
- l4irq\_wait\_any
  - Interface using direct functionality., [302](#)
- l4la\_alloc
  - list\_alloc.h, [2304](#)
- l4la\_avail
  - list\_alloc.h, [2304](#)
- l4la\_dump
  - list\_alloc.h, [2305](#)
- l4la\_free
  - list\_alloc.h, [2305](#)
- l4la\_init
  - list\_alloc.h, [2305](#)
- l4re\_aux\_t, [1505](#)
- l4re\_debug\_obj\_debug
  - Debug interface, [159](#)
- l4re\_dma\_space\_associate
  - DMA Space Interface, [151](#)
- l4re\_dma\_space\_direction
  - dma\_space.h, [1913](#)
- l4re\_dma\_space\_disassociate
  - DMA Space Interface, [151](#)
- l4re\_dma\_space\_dma\_addr\_t
  - dma\_space.h, [1913](#)
- l4re\_dma\_space\_map
  - DMA Space Interface, [152](#)
- l4re\_dma\_space\_space\_attrbs
  - dma\_space.h, [1914](#)
- l4re\_dma\_space\_t
  - DMA Space Interface, [150](#)
- l4re\_dma\_space\_unmap
  - DMA Space Interface, [153](#)
- l4re\_ds\_allocate
  - Dataspace interface, [155](#)
- l4re\_ds\_clear
  - Dataspace interface, [155](#)
- l4re\_ds\_copy\_in
  - Dataspace interface, [156](#)
- l4re\_ds\_flags
  - Dataspace interface, [156](#)
- l4re\_ds\_info
  - Dataspace interface, [156](#)
- l4re\_ds\_map\_flags
  - Dataspace interface, [155](#)
- l4re\_ds\_phys
  - Dataspace interface, [157](#)
- l4re\_ds\_size
  - Dataspace interface, [157](#)
- l4re\_ds\_stats\_t, [1506](#)
- l4re\_elf\_aux\_mword\_t, [1506](#)
- l4re\_elf\_aux\_t, [1507](#)
- l4re\_elf\_aux\_vma\_t, [1508](#)
- l4re\_env
  - Initial Environment, [286](#)
- l4re\_env\_cap\_entry\_t, [1509](#)
  - flags, [1510](#)
  - l4re\_env\_cap\_entry\_t, [1509](#)
- l4re\_env\_get\_cap
  - Initial Environment, [286](#)
- l4re\_env\_get\_cap\_e
  - Initial Environment, [287](#)
- l4re\_env\_get\_cap\_l
  - Initial Environment, [288](#)
- l4re\_env\_t, [1511](#)
  - env.h, [1967](#)
- l4re\_event\_get\_axis\_info
  - Event interface, [203](#)
- l4re\_event\_get\_buffer
  - Event interface, [204](#)
- l4re\_event\_get\_num\_streams
  - Event interface, [204](#)
- l4re\_event\_get\_stream\_info
  - Event interface, [205](#)
- l4re\_event\_get\_stream\_info\_for\_id
  - Event interface, [205](#)
- l4re\_event\_t, [1512](#)
- l4re\_inhibitor\_acquire
  - L4Re C Interface, [370](#)
- l4re\_kip
  - Initial Environment, [289](#)
- l4re\_log\_print
  - Log interface, [400](#)



- l4re\_log\_print\_srv
  - Log interface, [401](#)
- l4re\_log\_printn
  - Log interface, [402](#)
- l4re\_log\_printn\_srv
  - Log interface, [402](#)
- l4re\_ma\_alloc
  - Memory allocator, [408](#)
- l4re\_ma\_alloc\_align
  - Memory allocator, [409](#)
- l4re\_ma\_alloc\_align\_srv
  - Memory allocator, [410](#)
- l4re\_ma\_flags
  - Memory allocator, [407](#)
- l4re\_ma\_free
  - Memory allocator, [411](#)
- l4re\_ma\_free\_srv
  - Memory allocator, [412](#)
- l4re\_ns\_query\_srv
  - Namespace interface, [450](#)
- l4re\_ns\_query\_to\_srv
  - Namespace interface, [450](#)
- l4re\_ns\_register\_flags
  - Namespace interface, [449](#)
- l4re\_ns\_register\_obj\_srv
  - Namespace interface, [452](#)
- L4re\_protocols
  - protocols.h, [1999](#)
- l4re\_rm\_attach
  - Region map interface, [481](#)
- l4re\_rm\_attach\_srv
  - Region map interface, [482](#)
- l4re\_rm\_detach
  - Region map interface, [482](#)
- l4re\_rm\_detach\_ds
  - Region map interface, [483](#)
- l4re\_rm\_detach\_ds\_unmap
  - Region map interface, [484](#)
- l4re\_rm\_detach\_srv
  - Region map interface, [485](#)
- l4re\_rm\_detach\_unmap
  - Region map interface, [486](#)
- l4re\_rm\_find
  - Region map interface, [487](#)
- l4re\_rm\_find\_srv
  - Region map interface, [487](#)
- l4re\_rm\_flags\_t
  - Region map interface, [480](#)
- l4re\_rm\_free\_area
  - Region map interface, [488](#)
- l4re\_rm\_free\_area\_srv
  - Region map interface, [488](#)
- l4re\_rm\_reserve\_area
  - Region map interface, [489](#)
- l4re\_rm\_reserve\_area\_srv
  - Region map interface, [489](#)
- l4re\_rm\_show\_lists
  - Region map interface, [490](#)
- l4re\_util\_cap\_last
  - Capability allocator, [132](#)
- l4re\_util\_kumem\_alloc
  - kumem\_alloc.h, [1932](#)
- l4re\_video\_color\_component\_t, [1513](#)
- l4re\_video\_goos\_create\_buffer
  - Video API, [596](#)
- l4re\_video\_goos\_create\_view
  - Video API, [597](#)
- l4re\_video\_goos\_delete\_buffer
  - Video API, [597](#)
- l4re\_video\_goos\_delete\_view
  - Video API, [599](#)
- l4re\_video\_goos\_get\_static\_buffer
  - Video API, [599](#)
- l4re\_video\_goos\_get\_view
  - Video API, [599](#)
- l4re\_video\_goos\_info
  - Video API, [600](#)
- l4re\_video\_goos\_info\_flags\_t
  - Video API, [596](#)
- l4re\_video\_goos\_info\_t, [1514](#)
- l4re\_video\_goos\_refresh
  - Video API, [600](#)
- l4re\_video\_pixel\_info\_t, [1515](#)
- l4re\_video\_view\_get\_info
  - Video API, [601](#)
- l4re\_video\_view\_info\_flags\_t
  - Video API, [596](#)
- l4re\_video\_view\_info\_t, [1516](#)
- l4re\_video\_view\_refresh
  - Video API, [601](#)
- l4re\_video\_view\_set\_info
  - Video API, [601](#)
- l4re\_video\_view\_set\_viewport
  - Video API, [602](#)
- l4re\_video\_view\_stack
  - Video API, [602](#)
- l4re\_video\_view\_t, [1518](#)
  - Video API, [595](#)
- l4shmc\_add\_chunk
  - Chunks, [133](#)
- l4shmc\_add\_signal
  - Signals, [511](#)
- l4shmc\_area\_overhead
  - Shared Memory Library, [500](#)
- l4shmc\_area\_size
  - Shared Memory Library, [500](#)
- l4shmc\_area\_size\_free
  - Shared Memory Library, [501](#)
- l4shmc\_attach
  - Shared Memory Library, [501](#)
- l4shmc\_attach\_signal
  - Signals, [512](#)
- l4shmc\_attach\_signal\_to
  - Signals, [512](#)
- l4shmc\_attach\_to
  - Shared Memory Library, [502](#)

- l4shmc\_check\_magic
  - Signals, [513](#)
- l4shmc\_chunk\_capacity
  - Chunks, [134](#)
- l4shmc\_chunk\_consumed
  - Consumer, [141](#)
- l4shmc\_chunk\_overhead
  - Shared Memory Library, [502](#)
- l4shmc\_chunk\_ptr
  - Chunks, [134](#)
- l4shmc\_chunk\_ready
  - Producer, [472](#)
- l4shmc\_chunk\_ready\_sig
  - Producer, [473](#)
- l4shmc\_chunk\_signal
  - Chunks, [135](#)
- l4shmc\_chunk\_size
  - Consumer, [142](#)
- l4shmc\_chunk\_try\_to\_take
  - Producer, [473](#)
- l4shmc\_connect\_chunk\_signal
  - Shared Memory Library, [502](#)
- l4shmc\_create
  - Shared Memory Library, [504](#)
- l4shmc\_enable\_chunk
  - Consumer, [142](#)
- l4shmc\_enable\_signal
  - Consumer, [145](#)
- l4shmc\_get\_chunk
  - Chunks, [135](#)
- l4shmc\_get\_chunk\_to
  - Chunks, [136](#)
- l4shmc\_get\_signal\_to
  - Signals, [513](#)
- l4shmc\_is\_chunk\_clear
  - Producer, [473](#)
- l4shmc\_is\_chunk\_ready
  - Consumer, [143](#)
- l4shmc\_iterate\_chunk
  - Chunks, [136](#)
- l4shmc\_signal\_cap
  - Signals, [514](#)
- l4shmc\_trigger
  - Producer, [471](#)
- l4shmc\_wait\_any
  - Consumer, [146](#)
- l4shmc\_wait\_any\_to
  - Consumer, [146](#)
- l4shmc\_wait\_any\_try
  - Consumer, [146](#)
- l4shmc\_wait\_chunk
  - Consumer, [143](#)
- l4shmc\_wait\_chunk\_to
  - Consumer, [143](#)
- l4shmc\_wait\_chunk\_try
  - Consumer, [144](#)
- l4shmc\_wait\_signal
  - Consumer, [147](#)
- l4shmc\_wait\_signal\_to
  - Consumer, [147](#)
- l4shmc\_wait\_signal\_try
  - Consumer, [148](#)
- l4sigma0\_debug\_dump
  - Sigma0 API, [506](#)
- l4sigma0\_map\_anypage
  - Sigma0 API, [506](#)
- l4sigma0\_map\_errstr
  - Sigma0 API, [507](#)
- l4sigma0\_map\_iomem
  - Sigma0 API, [508](#)
- l4sigma0\_map\_kip
  - Sigma0 API, [508](#)
- l4sigma0\_map\_mem
  - Sigma0 API, [509](#)
- l4sigma0\_map\_tbuf
  - Sigma0 API, [509](#)
- l4sigma0\_new\_client
  - Sigma0 API, [510](#)
- l4sigma0\_return\_flags\_t
  - Sigma0 API, [506](#)
- l4util\_add8
  - Atomic Instructions, [94](#)
- l4util\_add8\_res
  - Atomic Instructions, [94](#)
- l4util\_atomic\_add
  - Atomic Instructions, [94](#)
- l4util\_atomic\_inc
  - Atomic Instructions, [95](#)
- l4util\_backtrace
  - backtrace.h, [2264](#)
- l4util\_bsf
  - Bit Manipulation, [109](#)
- l4util\_bsr
  - Bit Manipulation, [110](#)
- l4util\_btc
  - Bit Manipulation, [111](#)
- l4util\_btr
  - Bit Manipulation, [111](#)
- l4util\_bts
  - Bit Manipulation, [112](#)
- l4util\_clear\_bit
  - Bit Manipulation, [113](#)
- l4util\_cmpxchg
  - Atomic Instructions, [95](#)
- l4util\_cmpxchg16
  - Atomic Instructions, [96](#)
- l4util\_cmpxchg32
  - Atomic Instructions, [96](#)
- l4util\_cmpxchg64
  - Atomic Instructions, [98](#)
- l4util\_cmpxchg8
  - Atomic Instructions, [98](#)
- l4util\_complement\_bit
  - Bit Manipulation, [113](#)
- l4util\_cpu\_capabilities
  - CPU related functions, [121](#)

- l4util\_cpu\_capabilities\_nocheck
  - CPU related functions, [122](#)
- l4util\_cpu\_has\_cpuid
  - CPU related functions, [123](#)
- l4util\_find\_first\_set\_bit
  - Bit Manipulation, [114](#)
- l4util\_find\_first\_zero\_bit
  - Bit Manipulation, [114](#)
- l4util\_idt\_desc\_t, [1519](#)
- l4util\_idt\_header\_t, [1520](#)
- l4util\_in16
  - IA32 Port I/O API, [251](#)
- l4util\_in32
  - IA32 Port I/O API, [253](#)
- l4util\_in8
  - IA32 Port I/O API, [253](#)
- l4util\_inc8
  - Atomic Instructions, [99](#)
- l4util\_inc8\_res
  - Atomic Instructions, [99](#)
- l4util\_ins16
  - IA32 Port I/O API, [254](#)
- l4util\_ins32
  - IA32 Port I/O API, [254](#)
- l4util\_ins8
  - IA32 Port I/O API, [255](#)
- l4util\_ioport\_map
  - x86/l4/l4/util/port\_io.h, [1774](#)
- l4util\_irq\_acknowledge
  - amd64/l4/util/irq.h, [1897](#)
  - x86/l4/util/irq.h, [1900](#)
- l4util\_kip\_for\_each\_feature
  - Kernel Interface Page API, [332](#)
- l4util\_kip\_kernel\_abi\_version
  - Kernel Interface Page API, [333](#)
- l4util\_kip\_kernel\_has\_feature
  - Kernel Interface Page API, [333](#)
- l4util\_kip\_kernel\_is\_ux
  - Kernel Interface Page API, [333](#)
- l4util\_mb\_addr\_range\_t, [1521](#)
- l4util\_mb\_apm\_t, [1522](#)
- l4util\_mb\_drive\_t, [1522](#)
  - drive\_cylinders, [1523](#)
  - drive\_mode, [1523](#)
  - drive\_number, [1524](#)
- l4util\_mb\_info\_t, [1524](#)
- l4util\_mb\_mod\_t, [1526](#)
  - mod\_end, [1527](#)
  - mod\_start, [1527](#)
- l4util\_mb\_vbe\_ctrl\_t, [1527](#)
- l4util\_mb\_vbe\_mode\_t, [1528](#)
- l4util\_memdesc\_vm\_high
  - Kernel Interface Page API, [333](#)
- l4util\_micros2l4to
  - amd64/l4/util/util.h, [1754](#)
  - Utility Functions, [572](#)
  - x86/l4/util/util.h, [1757](#)
- l4util\_next\_power2
  - Bit Manipulation, [114](#)
- l4util\_out16
  - IA32 Port I/O API, [255](#)
- l4util\_out32
  - IA32 Port I/O API, [255](#)
- l4util\_out8
  - IA32 Port I/O API, [256](#)
- l4util\_outs16
  - IA32 Port I/O API, [256](#)
- l4util\_outs32
  - IA32 Port I/O API, [257](#)
- l4util\_outs8
  - IA32 Port I/O API, [257](#)
- l4util\_rand
  - Random number support, [475](#)
- l4util\_set\_bit
  - Bit Manipulation, [115](#)
- l4util\_splitlog2\_hdl
  - Utility Functions, [572](#)
- l4util\_splitlog2\_size
  - Utility Functions, [573](#)
- l4util\_srand
  - Random number support, [475](#)
- l4util\_stack\_get\_sp
  - amd64/l4/util/stack\_impl.h, [1812](#)
  - arm/l4/util/stack\_impl.h, [1811](#)
  - stack.h, [2325](#)
  - x86/l4/util/stack\_impl.h, [1814](#)
- l4util\_test\_bit
  - Bit Manipulation, [115](#)
- l4util\_xchg
  - Atomic Instructions, [99](#)
- l4util\_xchg16
  - Atomic Instructions, [100](#)
- l4util\_xchg32
  - Atomic Instructions, [100](#)
- l4util\_xchg8
  - Atomic Instructions, [101](#)
- L4vbus GPIO functions, [383](#)
  - l4vbus\_gpio\_config\_get, [385](#)
  - l4vbus\_gpio\_config\_pad, [385](#)
  - l4vbus\_gpio\_config\_pull, [386](#)
  - L4vbus\_gpio\_generic\_func, [384](#)
  - l4vbus\_gpio\_get, [387](#)
  - l4vbus\_gpio\_multi\_config\_pad, [387](#)
  - l4vbus\_gpio\_multi\_get, [388](#)
  - l4vbus\_gpio\_multi\_set, [389](#)
  - l4vbus\_gpio\_multi\_setup, [390](#)
  - L4vbus\_gpio\_pull\_modes, [384](#)
  - l4vbus\_gpio\_set, [391](#)
  - l4vbus\_gpio\_setup, [391](#)
  - l4vbus\_gpio\_to\_irq, [392](#)
- L4vbus PCI functions, [394](#)
  - l4vbus\_pci\_cfg\_read, [394](#)
  - l4vbus\_pci\_cfg\_write, [395](#)
  - l4vbus\_pci\_irq\_enable, [396](#)
  - l4vbus\_pciddev\_cfg\_read, [397](#)
  - l4vbus\_pciddev\_cfg\_write, [397](#)

- l4vbus\_pciddev\_irq\_enable, 398
- L4vbus::Device, 1531
  - \_bus, 1539
  - bus\_cap, 1533
  - dev\_handle, 1534
  - device, 1534
  - device\_by\_hid, 1536
  - get\_resource, 1537
  - is\_compatible, 1537
  - next\_device, 1538
  - operator!=, 1539
  - operator==, 1539
- L4vbus::Gpio\_module, 1540
  - config\_pad, 1543
  - get, 1544
  - pin, 1544
  - set, 1545
  - setup, 1545
- L4vbus::Gpio\_module::Pin\_slice, 1546
- L4vbus::Gpio\_pin, 1547
  - config\_get, 1550
  - config\_pad, 1551
  - config\_pull, 1551
  - get, 1552
  - pin, 1552
  - set, 1553
  - setup, 1553
  - to\_irq, 1554
- L4vbus::Icu, 1555
- L4vbus::Pci\_dev, 1557
  - cfg\_read, 1560
  - cfg\_write, 1560
  - irq\_enable, 1561
- L4vbus::Pci\_host\_bridge, 1562
  - cfg\_read, 1565
  - cfg\_write, 1566
  - irq\_enable, 1567
- L4vbus::Pm< DEC >, 1568
- L4vbus::Vbus, 1569
  - assign\_dma\_domain, 1572, 1573
  - release\_resource, 1574
  - request\_resource, 1574
  - root, 1575
- l4vbus\_assign\_dma\_domain
  - L4 Vbus functions, 358
- l4vbus\_device\_flags\_t
  - vbus\_types.h, 2347
- l4vbus\_device\_t, 1576
- L4vbus\_dma\_domain\_assign\_flags
  - L4 Vbus functions, 358
- l4vbus\_get\_device
  - L4 Vbus functions, 359
- l4vbus\_get\_device\_by\_hid
  - L4 Vbus functions, 360
- l4vbus\_get\_hid
  - L4 Vbus functions, 361
- l4vbus\_get\_next\_device
  - L4 Vbus functions, 361
- l4vbus\_get\_resource
  - L4 Vbus functions, 362
- l4vbus\_gpio\_config\_get
  - L4vbus GPIO functions, 385
- l4vbus\_gpio\_config\_pad
  - L4vbus GPIO functions, 385
- l4vbus\_gpio\_config\_pull
  - L4vbus GPIO functions, 386
- L4vbus\_gpio\_generic\_func
  - L4vbus GPIO functions, 384
- l4vbus\_gpio\_get
  - L4vbus GPIO functions, 387
- l4vbus\_gpio\_multi\_config\_pad
  - L4vbus GPIO functions, 387
- l4vbus\_gpio\_multi\_get
  - L4vbus GPIO functions, 388
- l4vbus\_gpio\_multi\_set
  - L4vbus GPIO functions, 389
- l4vbus\_gpio\_multi\_setup
  - L4vbus GPIO functions, 390
- L4vbus\_gpio\_pull\_modes
  - L4vbus GPIO functions, 384
- l4vbus\_gpio\_set
  - L4vbus GPIO functions, 391
- l4vbus\_gpio\_setup
  - L4vbus GPIO functions, 391
- l4vbus\_gpio\_to\_irq
  - L4vbus GPIO functions, 392
- l4vbus\_iface\_type\_t
  - vbus\_interfaces.h, 2344
- l4vbus\_is\_compatible
  - L4 Vbus functions, 363
- l4vbus\_pci\_cfg\_read
  - L4vbus PCI functions, 394
- l4vbus\_pci\_cfg\_write
  - L4vbus PCI functions, 395
- l4vbus\_pci\_irq\_enable
  - L4vbus PCI functions, 396
- l4vbus\_pciddev\_cfg\_read
  - L4vbus PCI functions, 397
- l4vbus\_pciddev\_cfg\_write
  - L4vbus PCI functions, 397
- l4vbus\_pciddev\_irq\_enable
  - L4vbus PCI functions, 398
- l4vbus\_release\_resource
  - L4 Vbus functions, 363
- l4vbus\_request\_resource
  - L4 Vbus functions, 364
- l4vbus\_resource\_t, 1577
- l4vbus\_resource\_type\_t
  - vbus\_types.h, 2347
- l4vbus\_subinterface\_supported
  - vbus\_interfaces.h, 2344
- l4vbus\_vicu\_get\_cap
  - L4 Vbus functions, 365
- L4vcpu::State, 1578
  - add, 1579
  - clear, 1579

- set, [1579](#)
- State, [1578](#)
- L4vcpu::Vcpu, [1580](#)
  - cast, [1584](#)
  - entry\_ip, [1584](#)
  - entry\_sp, [1585](#)
  - ext\_alloc, [1585](#)
  - i, [1586](#)
  - irq\_disable\_save, [1586](#)
  - irq\_enable, [1587](#)
  - irq\_restore, [1588](#)
  - is\_irq\_entry, [1588](#)
  - is\_page\_fault\_entry, [1589](#)
  - r, [1589](#), [1590](#)
  - saved\_state, [1590](#)
  - state, [1591](#)
  - task, [1591](#)
  - wait\_for\_event, [1592](#)
- l4vcpu\_ext\_alloc
  - Extended vCPU support, [210](#)
- l4vcpu\_irq\_disable
  - vCPU Support Library, [627](#)
- l4vcpu\_irq\_disable\_save
  - vCPU Support Library, [627](#)
- l4vcpu\_irq\_enable
  - vCPU Support Library, [628](#)
- l4vcpu\_irq\_restore
  - vCPU Support Library, [629](#)
- l4vcpu\_is\_irq\_entry
  - vCPU Support Library, [630](#)
- l4vcpu\_is\_page\_fault\_entry
  - vCPU Support Library, [631](#)
- l4vcpu\_print\_state
  - vCPU Support Library, [631](#)
- l4vcpu\_wait\_for\_event
  - vCPU Support Library, [632](#)
- L4virtio, [693](#)
- L4virtio::Device, [1593](#)
  - config\_queue, [1595](#)
  - device\_config, [1596](#)
  - device\_notification\_irq, [1596](#)
  - register\_ds, [1596](#)
  - register\_iface, [1597](#)
  - set\_status, [1597](#)
- L4virtio::Driver::Virtqueue, [1598](#)
  - alloc\_descriptor, [1601](#)
  - desc, [1601](#)
  - enqueue\_descriptor, [1602](#)
  - find\_next\_used, [1602](#)
  - free\_descriptor, [1602](#)
  - init\_queue, [1603](#)
  - initialize\_rings, [1604](#)
- L4virtio::Ptr
  - get, [1606](#)
  - Invalid\_type, [1606](#)
  - is\_valid, [1607](#)
- L4virtio::Ptr< T >, [1604](#)
- L4virtio::Svr::Bad\_descriptor, [1608](#)
  - Bad\_descriptor, [1609](#)
  - Error, [1609](#)
  - message, [1610](#)
- L4virtio::Svr::Block\_dev
  - Block\_dev, [1614](#)
  - finalize\_request, [1614](#)
  - get\_writeback, [1614](#)
  - process\_request, [1615](#)
  - register\_obj, [1615](#)
  - reset\_client, [1616](#)
  - set\_blk\_size, [1616](#)
  - set\_config\_wce, [1617](#)
  - set\_discard, [1617](#)
  - set\_flush, [1617](#)
  - set\_size\_max, [1618](#)
  - set\_topology, [1618](#)
  - set\_write\_zeroes, [1618](#)
- L4virtio::Svr::Block\_dev< Ds\_data >, [1610](#)
- L4virtio::Svr::Block\_request
  - data\_size, [1620](#)
  - next\_block, [1621](#)
- L4virtio::Svr::Block\_request< Ds\_data >, [1619](#)
- L4virtio::Svr::Data\_buffer, [1621](#)
  - copy\_to, [1623](#)
  - Data\_buffer, [1623](#)
  - done, [1624](#)
  - set, [1624](#)
  - skip, [1625](#)
- L4virtio::Svr::Dev\_config, [1626](#)
  - change\_queue\_config, [1628](#)
  - Dev\_config, [1627](#)
  - ds, [1629](#)
  - get\_cmd, [1629](#)
  - guest\_features, [1630](#)
  - hdr, [1631](#)
  - negotiated\_features, [1631](#)
  - qconfig, [1632](#)
  - reset\_cmd, [1632](#)
  - reset\_queue, [1633](#)
  - set\_failed, [1634](#)
  - set\_status, [1635](#)
  - status, [1636](#)
- L4virtio::Svr::Dev\_features, [1637](#)
  - ring\_event\_idx, [1639](#)
  - ring\_event\_idx\_bfm\_t, [1638](#)
  - ring\_indirect\_desc, [1639](#)
  - ring\_indirect\_desc\_bfm\_t, [1638](#)
- L4virtio::Svr::Dev\_status, [1640](#)
  - acked, [1642](#), [1643](#)
  - acked\_bfm\_t, [1641](#)
  - driver, [1643](#)
  - driver\_bfm\_t, [1641](#)
  - driver\_ok, [1644](#)
  - driver\_ok\_bfm\_t, [1642](#)
  - failed, [1644](#), [1645](#)
  - failed\_bfm\_t, [1642](#)
  - features\_ok, [1645](#)
  - features\_ok\_bfm\_t, [1642](#)

- running, 1645
- L4virtio::Svr::Device\_t
  - device\_error, 1649
  - device\_notify\_irq, 1649
  - handle\_mem\_cmd\_write, 1650
  - init\_mem\_info, 1650
  - register\_driver\_irq, 1650
  - reset\_queue\_config, 1651
  - setup\_queue, 1651
- L4virtio::Svr::Device\_t< DATA >, 1646
- L4virtio::Svr::Driver\_mem\_list\_t
  - add, 1654
  - find, 1655
  - full, 1655
  - init, 1656
  - load\_desc, 1657, 1658
  - remove, 1658
- L4virtio::Svr::Driver\_mem\_list\_t< DATA >, 1652
- L4virtio::Svr::Driver\_mem\_region\_t
  - contains, 1662
  - Driver\_mem\_region\_t, 1662
  - drv\_base, 1663
  - ds, 1663
  - ds\_offset, 1663
  - empty, 1663
  - flags, 1664
  - is\_writable, 1664
  - local, 1664
  - local\_base, 1665
  - size, 1665
- L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 1659
- L4virtio::Svr::Request\_processor, 1666
  - current\_flags, 1667
  - has\_more, 1667
  - next, 1668
  - start, 1670, 1671
- L4virtio::Svr::Virtqueue, 1672
  - consumed, 1675
  - desc, 1676
  - desc\_avail, 1676
  - disable\_notify, 1677
  - enable\_notify, 1677
  - next\_avail, 1677
- L4virtio::Svr::Virtqueue::Head\_desc, 1679
  - desc, 1679
  - operator Null\_ptr\_check const \*, 1680
  - valid, 1680
- L4virtio::Virtqueue, 1681
  - avail\_align, 1684
  - avail\_size, 1684
  - desc\_align, 1685
  - desc\_size, 1685
  - disable, 1686
  - dump, 1686
  - get\_avail\_idx, 1687
  - get\_tail\_avail\_idx, 1687
  - no\_notify\_guest, 1688
  - no\_notify\_host, 1688, 1689
  - num, 1689
  - ready, 1690
  - setup, 1690
  - setup\_simple, 1691
  - total\_size, 1692
  - used\_align, 1693
  - used\_size, 1693
- L4virtio::Virtqueue::Avail, 1694
- L4virtio::Virtqueue::Avail::Flags, 1695
  - no\_irq, 1696
  - no\_irq\_bfm\_t, 1696
- L4virtio::Virtqueue::Desc, 1697
- L4virtio::Virtqueue::Desc::Flags, 1699
  - indirect, 1701
  - indirect\_bfm\_t, 1700
  - next, 1702
  - next\_bfm\_t, 1701
  - write, 1702
  - write\_bfm\_t, 1701
- L4virtio::Virtqueue::Used, 1703
- L4virtio::Virtqueue::Used::Flags, 1704
  - no\_notify, 1705, 1706
  - no\_notify\_bfm\_t, 1705
- L4virtio::Virtqueue::Used\_elem, 1706
  - Used\_elem, 1707
- I4virtio\_block\_config\_t, 1707
  - blk\_size, 1708
- I4virtio\_block\_discard\_t, 1709
- I4virtio\_block\_header\_t, 1710
- L4virtio\_block\_operations
  - L4 VIRTIO Block Device, 344
- L4virtio\_block\_status
  - L4 VIRTIO Block Device, 345
- I4virtio\_config\_hdr\_t, 1711
  - magic, 1712
  - status, 1712
- I4virtio\_config\_queue
  - L4 VIRTIO Transport Layer, 351
- I4virtio\_config\_queue\_t, 1713
  - L4 VIRTIO Transport Layer, 349
- I4virtio\_config\_queues
  - L4 VIRTIO Transport Layer, 352
- I4virtio\_device\_config
  - L4 VIRTIO Transport Layer, 353
- I4virtio\_device\_config\_ds
  - L4 VIRTIO Transport Layer, 353
- L4virtio\_device\_ids
  - L4 VIRTIO Transport Layer, 350
- I4virtio\_device\_notification\_irq
  - L4 VIRTIO Transport Layer, 354
- L4virtio\_device\_status
  - L4 VIRTIO Transport Layer, 351
- L4virtio\_feature\_bits
  - L4 VIRTIO Transport Layer, 351
- I4virtio\_register\_ds
  - L4 VIRTIO Transport Layer, 354
- I4virtio\_register\_iface
  - L4 VIRTIO Transport Layer, 355

- l4virtio\_set\_status
  - L4 VIRTIO Transport Layer, [356](#)
- length
  - L4::lpc::Varg, [1022](#)
- libedid\_check\_header
  - EDID parsing functionality, [162](#)
- libedid\_checksum
  - EDID parsing functionality, [162](#)
- Libedid\_consts
  - EDID parsing functionality, [161](#)
- libedid\_dump
  - EDID parsing functionality, [162](#)
- libedid\_dump\_standard\_timings
  - EDID parsing functionality, [163](#)
- libedid\_num\_ext\_blocks
  - EDID parsing functionality, [163](#)
- libedid\_pnp\_id
  - EDID parsing functionality, [163](#)
- libedid\_prefered\_resolution
  - EDID parsing functionality, [164](#)
- libedid\_revision
  - EDID parsing functionality, [164](#)
- libedid\_version
  - EDID parsing functionality, [164](#)
- link
  - L4Re::Vfs::Directory, [1449](#)
- List\_alloc
  - cxx::List\_alloc, [808](#)
- list\_alloc.h
  - l4la\_alloc, [2304](#)
  - l4la\_avail, [2304](#)
  - l4la\_dump, [2305](#)
  - l4la\_free, [2305](#)
  - l4la\_init, [2305](#)
- load\_desc
  - L4virtio::Svr::Driver\_mem\_list\_t, [1657](#), [1658](#)
- local
  - L4virtio::Svr::Driver\_mem\_region\_t, [1664](#)
- local\_base
  - L4virtio::Svr::Driver\_mem\_region\_t, [1665](#)
- local\_id\_received
  - L4::lpc::Gen\_fpage, [968](#)
- log
  - L4Re::Env, [1316](#)
- Log interface, [400](#)
  - l4re\_log\_print, [400](#)
  - l4re\_log\_print\_srv, [401](#)
  - l4re\_log\_printh, [402](#)
  - l4re\_log\_printh\_srv, [402](#)
- Logging interface, [404](#)
- loop
  - L4::Server, [1151](#)
- Low-Level Thread Functions, [405](#)
- lower\_bound\_node
  - cxx::Bits::Base\_avl\_set, [764](#)
  - cxx::Bits::Bst, [780](#)
- lseek64
  - L4Re::Vfs::Regular\_file, [1473](#)
- Lstr
  - L4::Factory::Lstr, [925](#)
- MB\_ARD\_MEMORY
  - mb\_info.h, [2310](#)
- MB\_ART\_MEMORY
  - mb\_info.h, [2311](#)
- Machine Restarting Function, [406](#)
- magic
  - l4\_vhw\_descriptor, [1276](#)
  - l4virtio\_config\_hdr\_t, [1712](#)
- main\_thread
  - L4Re::Env, [1316](#), [1317](#)
- make\_auto\_cap
  - L4Re Capability API, [375](#)
- make\_auto\_del\_cap
  - L4Re Capability API, [375](#)
- make\_cap
  - L4::lpc, [655](#)
- make\_cap\_full
  - L4::lpc, [656](#)
- make\_cap\_rw
  - L4::lpc, [657](#)
- make\_cap\_rws
  - L4::lpc, [658](#)
- make\_ref\_cap
  - L4Re Capability API, [375](#)
- make\_ref\_del\_cap
  - L4Re Capability API, [376](#)
- make\_shared\_cap
  - L4Re, [679](#)
  - L4Re::Util, [690](#)
- make\_shared\_del\_cap
  - L4Re, [680](#)
  - L4Re::Util, [690](#)
- make\_unique\_cap
  - L4Re, [680](#)
  - L4Re::Util, [691](#)
- make\_unique\_del\_cap
  - L4Re, [681](#)
  - L4Re::Util, [691](#)
- map
  - L4::Task, [1170](#)
  - L4Re::Dataspace, [1297](#)
  - L4Re::Dma\_space, [1307](#)
  - L4Re::Util::Dataspace\_svr, [1396](#)
- Map\_flags
  - L4Re::Dataspace, [1293](#)
- map\_hook
  - L4Re::Util::Dataspace\_svr, [1397](#)
- map\_region
  - L4Re::Dataspace, [1298](#)
- mask
  - L4::lcu, [936](#)
- Masks
  - cxx::Bitfield, [726](#)
- max
  - Small C++ Template Library, [516](#)
- mb\_info.h



- L4UTIL\_MB\_MEMORY, [2310](#)
- MB\_ARD\_MEMORY, [2310](#)
- MB\_ART\_MEMORY, [2311](#)
- mem\_alloc
  - L4Re::Env, [1317](#)
- Mem\_alloc\_flags
  - L4Re::Mem\_alloc, [1340](#)
- Mem\_desc
  - L4::Kip::Mem\_desc, [1081](#)
- mem\_size
  - l4\_vhw\_entry, [1278](#)
- mem\_start
  - l4\_vhw\_entry, [1278](#)
- Mem\_type
  - L4::Kip::Mem\_desc, [1081](#)
- Memory allocator, [407](#)
  - l4re\_ma\_alloc, [408](#)
  - l4re\_ma\_alloc\_align, [409](#)
  - l4re\_ma\_alloc\_align\_srv, [410](#)
  - l4re\_ma\_flags, [407](#)
  - l4re\_ma\_free, [411](#)
  - l4re\_ma\_free\_srv, [412](#)
- Memory descriptors (C version), [413](#)
  - l4\_kernel\_info\_get\_mem\_desc\_end, [415](#)
  - l4\_kernel\_info\_get\_mem\_desc\_is\_virtual, [415](#)
  - l4\_kernel\_info\_get\_mem\_desc\_start, [415](#)
  - l4\_kernel\_info\_get\_mem\_desc\_subtype, [416](#)
  - l4\_kernel\_info\_get\_mem\_desc\_type, [416](#)
  - l4\_kernel\_info\_get\_num\_mem\_descs, [416](#)
  - l4\_kernel\_info\_mem\_desc\_t, [414](#)
  - l4\_kernel\_info\_set\_mem\_desc, [417](#)
  - l4\_mem\_info\_sub\_type\_t, [414](#)
  - l4\_mem\_type\_t, [415](#)
- Memory operations., [418](#)
  - L4\_mem\_op\_widths, [418](#)
  - l4\_mem\_read, [419](#)
  - l4\_mem\_write, [419](#)
- Memory related, [421](#)
  - L4\_LOG2\_PAGESIZE, [422](#)
  - L4\_LOG2\_SUPERPAGESIZE, [422](#)
  - L4\_PAGEMASK, [422](#)
  - L4\_SUPERPAGEMASK, [423](#)
  - L4\_SUPERPAGESIZE, [423](#)
  - l4\_addr\_consts\_t, [423](#)
  - l4\_round\_page, [424](#)
  - l4\_round\_size, [425](#)
  - l4\_trunc\_page, [426](#)
  - l4\_trunc\_size, [426](#)
- message
  - L4virtio::Svr::Bad\_descriptor, [1610](#)
- Message Items, [428](#)
  - l4\_map\_control, [429](#)
  - l4\_map\_obj\_control, [430](#)
  - l4\_msg\_item\_consts\_t, [428](#)
- Message Registers (MRs), [431](#)
- Message Tag, [432](#)
  - l4\_msgtag, [435](#)
  - l4\_msgtag\_flags, [434](#), [437](#)
  - l4\_msgtag\_has\_error, [438](#)
  - l4\_msgtag\_is\_exception, [439](#)
  - l4\_msgtag\_is\_io\_page\_fault, [440](#)
  - l4\_msgtag\_is\_page\_fault, [441](#)
  - l4\_msgtag\_is\_preemption, [442](#)
  - l4\_msgtag\_is\_sigma0, [443](#)
  - l4\_msgtag\_is\_sys\_exception, [444](#)
  - l4\_msgtag\_items, [445](#)
  - l4\_msgtag\_label, [446](#)
  - l4\_msgtag\_protocol, [434](#)
  - l4\_msgtag\_t, [433](#)
  - l4\_msgtag\_words, [447](#)
  - L4\_platform\_ctl\_proto, [435](#)
- min
  - Small C++ Template Library, [516](#)
- mkdir
  - L4Re::Vfs::Directory, [1449](#)
- mmio\_read
  - L4Re::Mmio\_space, [1345](#)
- mmio\_write
  - L4Re::Mmio\_space, [1346](#)
- mod\_end
  - l4util\_mb\_mod\_t, [1527](#)
- mod\_start
  - l4util\_mb\_mod\_t, [1527](#)
- Mode
  - L4Re::Util::Event\_t, [1412](#)
- modify\_senders
  - L4::Thread, [1178](#)
- mount
  - L4Re::Vfs::File\_system, [1455](#)
  - L4Re::Vfs::Fs, [1460](#)
- move
  - L4::Cap, [885](#)
  - L4::Cap\_base, [893](#)
- msg\_add
  - L4::lpc::Msg, [664](#)
- msg\_get
  - L4::lpc::Msg, [665](#)
- Msg\_ptr
  - L4::lpc::Msg\_ptr, [1006](#)
- msg\_ptr
  - L4::lpc, [658](#)
- msi\_data
  - l4\_icu\_msi\_info\_t, [1249](#)
- msi\_info
  - L4::lcu, [937](#)
- NT\_VERSION
  - ELF binary format, [189](#)
- Name-space API, [448](#)
- Namespace interface, [449](#)
  - l4re\_ns\_query\_srv, [450](#)
  - l4re\_ns\_query\_to\_srv, [450](#)
  - l4re\_ns\_register\_flags, [449](#)
  - l4re\_ns\_register\_obj\_srv, [452](#)
- negotiated\_features
  - L4virtio::Svr::Dev\_config, [1631](#)
- next



- L4Re::Event\_buffer\_t, [1327](#)
  - L4virtio::Svr::Request\_processor, [1668](#)
  - L4virtio::Virtqueue::Desc::Flags, [1702](#)
- next\_avail
  - L4virtio::Svr::Virtqueue, [1677](#)
- next\_bfm\_t
  - L4virtio::Virtqueue::Desc::Flags, [1701](#)
- next\_block
  - L4virtio::Svr::Block\_request, [1621](#)
- next\_device
  - L4vbus::Device, [1538](#)
- next\_lock\_info
  - L4Re::Inhibitor, [1333](#)
- next\_timeout
  - L4::lpc\_svr::Timeout\_queue, [1058](#)
- no\_demand
  - L4::Type\_info::Demand, [1195](#)
- No\_init\_type
  - L4::Cap\_base, [888](#)
- no\_irq
  - L4virtio::Virtqueue::Avail::Flags, [1696](#)
- no\_irq\_bfm\_t
  - L4virtio::Virtqueue::Avail::Flags, [1696](#)
- no\_notify
  - L4virtio::Virtqueue::Used::Flags, [1705](#), [1706](#)
- no\_notify\_bfm\_t
  - L4virtio::Virtqueue::Used::Flags, [1705](#)
- no\_notify\_guest
  - L4virtio::Virtqueue, [1688](#)
- no\_notify\_host
  - L4virtio::Virtqueue, [1688](#), [1689](#)
- None\_type
  - L4::Types::Flags, [1223](#)
- num
  - L4virtio::Virtqueue, [1689](#)
- num\_interfaces
  - L4::Meta, [1110](#)
- Object Invocation, [454](#)
  - l4\_ipc, [456](#)
  - l4\_ipc\_call, [457](#)
  - l4\_ipc\_receive, [459](#)
  - l4\_ipc\_reply\_and\_wait, [460](#)
  - l4\_ipc\_send, [461](#)
  - l4\_ipc\_send\_and\_wait, [462](#)
  - l4\_ipc\_sleep, [464](#)
  - l4\_ipc\_wait, [464](#)
  - l4\_sndfpage\_add, [466](#)
  - l4\_syscall\_flags\_t, [456](#)
- Object\_registry
  - L4Re::Util::Object\_registry, [1423](#)
- offset
  - l4\_sched\_cpu\_set\_t, [1256](#)
- Opcode
  - L4::Platform\_control, [1120](#)
- operator l4\_msgtag\_t
  - L4::Factory::S, [928](#)
- operator new
  - Small C++ Template Library, [517](#)
- operator Null\_ptr\_check const \*
  - L4virtio::Svr::Virtqueue::Head\_desc, [1680](#)
- operator Priv\_type \*
  - cxx::Auto\_ptr, [698](#)
- operator!
  - cxx::Bits::Direction, [788](#)
- operator!=
  - L4vbus::Device, [1539](#)
- operator<<
  - ipc\_stream, [1864](#)–[1866](#)
  - L4::Factory::S, [928](#)–[930](#)
- operator>>
  - ipc\_stream, [1867](#)–[1871](#)
- operator\*
  - cxx::Auto\_ptr, [698](#)
  - cxx::Bits::Base\_avl\_set::Node, [768](#)
- operator()
  - cxx::Pair\_first\_compare, [828](#)
- operator->
  - cxx::Auto\_ptr, [698](#)
  - cxx::Bits::Base\_avl\_set::Node, [768](#)
- operator=
  - cxx::Auto\_ptr, [698](#)
- operator==
  - L4Re::Video::Color\_component, [1480](#)
  - L4Re::Video::Pixel\_info, [1496](#)
  - L4vbus::Device, [1539](#)
- operator[]
  - cxx::Avl\_map, [703](#), [704](#)
  - cxx::Bitmap\_base, [748](#)
  - cxx::List, [805](#)
- PT\_GNU\_EH\_FRAME
  - ELF binary format, [190](#)
- PT\_GNU\_RELRO
  - ELF binary format, [190](#)
- PT\_GNU\_STACK
  - ELF binary format, [190](#)
- PT\_HIOS
  - ELF binary format, [190](#)
- PT\_HIPROC
  - ELF binary format, [190](#)
- PT\_L4\_AUX
  - ELF binary format, [191](#)
- PT\_L4\_KIP
  - ELF binary format, [191](#)
- PT\_L4\_STACK
  - ELF binary format, [191](#)
- PT\_LOOS
  - ELF binary format, [191](#)
- PT\_LOPROC
  - ELF binary format, [191](#)
- page\_fault
  - L4::Pager, [1117](#)
- page\_shift
  - L4Re::Util::Dataspace\_svr, [1398](#)
- pager
  - L4::Thread::Attr, [1185](#), [1186](#)
- Pair

- cxx::Pair, [827](#)
- Pair\_first\_compare
  - cxx::Pair\_first\_compare, [828](#)
- parent
  - L4Re::Env, [1318](#)
- Parent API, [467](#)
- parse\_cmdline
  - Comfortable Command Line Parsing, [137](#)
- phys
  - L4Re::Dataspace, [1299](#)
  - L4Re::Util::Dataspace\_svr, [1399](#)
- pin
  - L4vbus::Gpio\_module, [1544](#)
  - L4vbus::Gpio\_pin, [1552](#)
- Pixel\_info
  - L4Re::Video::Pixel\_info, [1491](#), [1492](#)
- Platform Control C API, [468](#)
  - l4\_platform\_ctl\_cpu\_disable, [468](#)
  - l4\_platform\_ctl\_cpu\_enable, [469](#)
  - l4\_platform\_ctl\_system\_shutdown, [469](#)
  - l4\_platform\_ctl\_system\_suspend, [470](#)
- Poll\_timeout\_kipclock
  - L4::Poll\_timeout\_kipclock, [1123](#)
- pop\_front
  - cxx::Bits::Smart\_ptr\_list, [790](#)
  - cxx::H\_list, [797](#)
  - cxx::S\_list, [839](#)
- print
  - L4Re::Log, [1337](#)
- println
  - L4Re::Log, [1337](#)
- process
  - L4Re::Util::Event\_buffer\_consumer\_t, [1404](#)
- process\_request
  - L4virtio::Svr::Block\_dev, [1615](#)
- Producer, [471](#), [472](#)
  - l4shmc\_chunk\_ready, [472](#)
  - l4shmc\_chunk\_ready\_sig, [473](#)
  - l4shmc\_chunk\_try\_to\_take, [473](#)
  - l4shmc\_is\_chunk\_clear, [473](#)
  - l4shmc\_trigger, [471](#)
- proto\_dispatch
  - L4::Kobject\_typeid, [1104](#)
  - L4::Server\_object\_t, [1157](#)
- protocols.h
  - L4re\_protocols, [1999](#)
- provider\_pid
  - l4\_vhw\_entry, [1279](#)
- ptr
  - cxx::Ref\_ptr, [835](#)
- push\_back
  - cxx::List, [805](#)
  - cxx::List\_item, [815](#)
- push\_front
  - cxx::List, [806](#)
  - cxx::List\_item, [816](#)
- put
  - L4::lpc::Ostream, [1011](#), [1012](#)
- L4Re::Event\_buffer\_t, [1327](#)
- qconfig
  - L4virtio::Svr::Dev\_config, [1632](#)
- query
  - L4Re::Namespace, [1350](#), [1351](#)
- query\_log\_name
  - L4::Debugger, [903](#)
- query\_log\_typeid
  - L4::Debugger, [904](#)
- Query\_result\_flags
  - L4Re::Namespace, [1349](#)
- r
  - L4Re::Video::Pixel\_info, [1497](#)
  - L4vcpu::Vcpu, [1589](#), [1590](#)
- Random number support, [475](#)
  - l4util\_rand, [475](#)
  - l4util\_srand, [475](#)
- rbegin
  - cxx::Bits::Base\_avl\_set, [764](#), [765](#)
  - cxx::Bits::Bst, [781](#), [782](#)
- rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1049](#), [1050](#)
- rcv\_endpoint.h
  - L4\_rcv\_ep\_ops, [2192](#)
- read
  - L4::lpc, [659](#)
  - L4::Vcon, [1235](#)
- read\_with\_flags
  - L4::Vcon, [1236](#)
- readv
  - L4Re::Vfs::Regular\_file, [1473](#)
- ready
  - L4virtio::Virtqueue, [1690](#)
- realloc\_rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1051](#)
  - L4Re::Util::Br\_manager, [1378](#)
- Realtime API, [477](#)
- receive
  - L4::lpc::Istream, [982](#)
  - L4::lrc, [1068](#)
- Ref
  - cxx::Bitfield, [724](#)
- Ref\_ptr
  - cxx::Ref\_ptr, [833](#), [834](#)
- Ref\_type
  - cxx::Auto\_ptr, [696](#)
- Ref\_unshifted
  - cxx::Bitfield, [725](#)
- refresh
  - L4Re::Util::Video::Goos\_svr, [1438](#)
  - L4Re::Video::View, [1500](#)
- Region map API, [478](#)
- Region map interface, [479](#)
  - l4re\_rm\_attach, [481](#)
  - l4re\_rm\_attach\_srv, [482](#)
  - l4re\_rm\_detach, [482](#)
  - l4re\_rm\_detach\_ds, [483](#)

- l4re\_rm\_detach\_ds\_unmap, [484](#)
- l4re\_rm\_detach\_srv, [485](#)
- l4re\_rm\_detach\_unmap, [486](#)
- l4re\_rm\_find, [487](#)
- l4re\_rm\_find\_srv, [487](#)
- l4re\_rm\_flags\_t, [480](#)
- l4re\_rm\_free\_area, [488](#)
- l4re\_rm\_free\_area\_srv, [488](#)
- l4re\_rm\_reserve\_area, [489](#)
- l4re\_rm\_reserve\_area\_srv, [489](#)
- l4re\_rm\_show\_lists, [490](#)
- Region\_flags
  - L4Re::Rm, [1362](#)
- register\_del\_irq
  - L4::Thread, [1179](#)
- register\_driver\_irq
  - L4virtio::Svr::Device\_t, [1650](#)
- register\_ds
  - L4virtio::Device, [1596](#)
- Register\_flags
  - L4Re::Namespace, [1349](#)
- register\_iface
  - L4virtio::Device, [1597](#)
- register\_irq\_obj
  - L4::Registry\_iface, [1132](#)
  - L4Re::Util::Object\_registry, [1424](#)
- register\_obj
  - L4::Registry\_iface, [1133](#), [1134](#)
  - L4Re::Namespace, [1352](#)
  - L4Re::Util::Object\_registry, [1425](#), [1426](#)
  - L4virtio::Svr::Block\_dev, [1615](#)
- registry
  - L4Re::Util::Registry\_server, [1432](#), [1433](#)
- Registry\_server
  - L4Re::Util::Registry\_server, [1432](#)
- release
  - cxx::Auto\_ptr, [699](#)
  - cxx::Ref\_ptr, [835](#)
  - L4Re::Inhibitor, [1334](#)
  - L4Re::Util::Counting\_cap\_alloc, [1389](#)
  - L4Re::Util::Dataspace\_svr, [1400](#)
- release\_cap
  - L4::Task, [1170](#)
- release\_resource
  - L4vbus::Vbus, [1574](#)
- remove
  - cxx::Avl\_tree, [711](#)
  - cxx::Bits::Base\_avl\_set, [765](#)
  - cxx::H\_list, [798](#)
  - cxx::List, [806](#)
  - cxx::List\_item, [816](#)
  - L4::lpc\_svr::Timeout\_queue, [1058](#)
  - L4virtio::Svr::Driver\_mem\_list\_t, [1658](#)
- remove\_all
  - cxx::Bits::Bst, [782](#)
- remove\_me
  - cxx::List\_item, [817](#)
  - cxx::List\_item::Iter, [820](#)
- remove\_timeout
  - L4::lpc\_svr::Server\_iface, [1052](#)
  - L4::lpc\_svr::Timeout\_queue\_hooks, [1062](#)
- remove\_tree
  - cxx::Bits::Bst, [783](#)
- rename
  - L4Re::Vfs::Directory, [1450](#)
- rend
  - cxx::Bits::Base\_avl\_set, [766](#)
  - cxx::Bits::Bst, [783](#), [784](#)
- replace
  - cxx::H\_list, [798](#)
- reply\_and\_wait
  - L4::lpc::loststream, [974](#)
- Reply\_mode
  - Server-Side IPC framework, [498](#)
- request\_resource
  - L4vbus::Vbus, [1574](#)
- reserve\_area
  - L4Re::Rm, [1368](#), [1369](#)
- reset
  - L4::lpc::loststream, [975](#)
  - L4::lpc::lstream, [983](#)
- reset\_client
  - L4virtio::Svr::Block\_dev, [1616](#)
- reset\_cmd
  - L4virtio::Svr::Dev\_config, [1632](#)
- reset\_queue
  - L4virtio::Svr::Dev\_config, [1633](#)
- reset\_queue\_config
  - L4virtio::Svr::Device\_t, [1651](#)
- ring\_event\_idx
  - L4virtio::Svr::Dev\_features, [1639](#)
- ring\_event\_idx\_bfm\_t
  - L4virtio::Svr::Dev\_features, [1638](#)
- ring\_indirect\_desc
  - L4virtio::Svr::Dev\_features, [1639](#)
- ring\_indirect\_desc\_bfm\_t
  - L4virtio::Svr::Dev\_features, [1638](#)
- rm
  - L4Re::Env, [1318](#), [1319](#)
- rmdir
  - L4Re::Vfs::Directory, [1450](#)
- root
  - L4vbus::Vbus, [1575](#)
- Rpcs
  - L4::Exception, [910](#)
- run\_thread
  - L4::Scheduler, [1143](#)
- running
  - L4virtio::Svr::Dev\_status, [1645](#)
- Runtime\_error
  - L4::Runtime\_error, [1138](#)
- S
  - L4::Factory::S, [927](#)
- SHF\_GROUP
  - ELF binary format, [192](#)
- SHF\_MASKOS

- ELF binary format, [192](#)
- SHF\_TLS
  - ELF binary format, [192](#)
- SHT\_NUM
  - ELF binary format, [192](#)
- saved\_state
  - L4vcpu::Vcpu, [1590](#)
- scan\_zero
  - cxx::Bitmap, [740](#)
  - cxx::Bitmap\_base, [749](#)
- Scheduler, [491](#)
  - l4\_sched\_cpu\_set, [492](#)
  - l4\_scheduler\_idle\_time, [493](#)
  - l4\_scheduler\_info, [494](#)
  - l4\_scheduler\_is\_online, [495](#)
  - l4\_scheduler\_ops, [492](#)
  - l4\_scheduler\_run\_thread, [495](#)
- scheduler
  - L4Re::Env, [1319](#)
- screen\_info
  - L4Re::Util::Video::Goos\_svr, [1439](#)
- send
  - L4::lpc::Ostream, [1012](#)
  - L4::Vcon, [1236](#)
- Server
  - L4::Server, [1150](#)
- Server-Side IPC framework, [497](#)
  - Reply\_mode, [498](#)
- set
  - cxx::Bitfield, [727](#)
  - L4::Kip::Mem\_desc, [1086](#)
  - L4::Poll\_timeout\_kipclock, [1123](#)
  - L4Re::Video::Color\_component, [1480](#)
  - L4vbus::Gpio\_module, [1545](#)
  - L4vbus::Gpio\_pin, [1553](#)
  - L4vcpu::State, [1579](#)
  - L4virtio::Svr::Data\_buffer, [1624](#)
- set\_attr
  - L4::Vcon, [1237](#)
- set\_bit
  - cxx::Bitmap\_base, [750](#)
- set\_blk\_size
  - L4virtio::Svr::Block\_dev, [1616](#)
- set\_config\_wce
  - L4virtio::Svr::Block\_dev, [1617](#)
- set\_dirty
  - cxx::Bitfield, [728](#)
- set\_discard
  - L4virtio::Svr::Block\_dev, [1617](#)
- set\_failed
  - L4virtio::Svr::Dev\_config, [1634](#)
- set\_fd
  - L4Re::Vfs::Fs, [1460](#)
- set\_flush
  - L4virtio::Svr::Block\_dev, [1617](#)
- set\_info
  - L4Re::Video::View, [1501](#)
- set\_lock
  - L4Re::Vfs::Regular\_file, [1474](#)
- set\_mode
  - L4::lcu, [937](#)
- set\_object\_name
  - L4::Debugger, [905](#)
- set\_rcv\_cap\_flags
  - L4Re::Util::Br\_manager, [1378](#)
- set\_size\_max
  - L4virtio::Svr::Block\_dev, [1618](#)
- set\_status
  - L4virtio::Device, [1597](#)
  - L4virtio::Svr::Dev\_config, [1635](#)
- set\_status\_flags
  - L4Re::Vfs::Generic\_file, [1464](#)
- set\_topology
  - L4virtio::Svr::Block\_dev, [1618](#)
- set\_unshifted
  - cxx::Bitfield, [729](#)
- set\_unshifted\_dirty
  - cxx::Bitfield, [730](#)
- set\_viewport
  - L4Re::Video::View, [1501](#)
- set\_write\_zeroes
  - L4virtio::Svr::Block\_dev, [1618](#)
- setup
  - L4Re::Util::Counting\_cap\_alloc, [1390](#)
  - L4vbus::Gpio\_module, [1545](#)
  - L4vbus::Gpio\_pin, [1553](#)
  - L4virtio::Virtqueue, [1690](#)
- setup\_queue
  - L4virtio::Svr::Device\_t, [1651](#)
- setup\_simple
  - L4virtio::Virtqueue, [1691](#)
- Shared Memory Library, [499](#)
  - l4shmc\_area\_overhead, [500](#)
  - l4shmc\_area\_size, [500](#)
  - l4shmc\_area\_size\_free, [501](#)
  - l4shmc\_attach, [501](#)
  - l4shmc\_attach\_to, [502](#)
  - l4shmc\_chunk\_overhead, [502](#)
  - l4shmc\_connect\_chunk\_signal, [502](#)
  - l4shmc\_create, [504](#)
- Shared\_cap
  - L4Re, [670](#)
  - L4Re::Util, [684](#)
- shared\_cap
  - L4Re, [670](#)
  - L4Re::Util, [685](#)
- Shared\_del\_cap
  - L4Re, [671](#)
  - L4Re::Util, [686](#)
- shared\_del\_cap
  - L4Re, [671](#)
  - L4Re::Util, [686](#)
- shift
  - L4Re::Video::Color\_component, [1480](#)
- Shift\_type
  - cxx::Bitfield, [725](#)

- si
  - l4\_vcpu\_regs\_t, [1271](#)
- Sigma0 API, [505](#)
  - l4sigma0\_debug\_dump, [506](#)
  - l4sigma0\_map\_anypage, [506](#)
  - l4sigma0\_map\_errstr, [507](#)
  - l4sigma0\_map\_iomem, [508](#)
  - l4sigma0\_map\_kip, [508](#)
  - l4sigma0\_map\_mem, [509](#)
  - l4sigma0\_map\_tbuf, [509](#)
  - l4sigma0\_new\_client, [510](#)
  - l4sigma0\_return\_flags\_t, [506](#)
- signal
  - L4Re::Parent, [1357](#)
- Signals, [511](#)
  - l4shmc\_add\_signal, [511](#)
  - l4shmc\_attach\_signal, [512](#)
  - l4shmc\_attach\_signal\_to, [512](#)
  - l4shmc\_check\_magic, [513](#)
  - l4shmc\_get\_signal\_to, [513](#)
  - l4shmc\_signal\_cap, [514](#)
- size
  - cxx::List, [806](#)
  - L4::Kip::Mem\_desc, [1086](#)
  - L4Re::Dataspace, [1300](#)
  - L4Re::Video::Color\_component, [1481](#)
  - L4virtio::Svr::Driver\_mem\_region\_t, [1665](#)
- skip
  - L4::lpc::Istream, [983](#)
  - L4virtio::Svr::Data\_buffer, [1625](#)
- Small C++ Template Library, [515](#)
  - max, [516](#)
  - min, [516](#)
  - operator new, [517](#)
- Small\_buf
  - L4::lpc::Small\_buf, [1017](#)
- Smart\_cap
  - L4::Smart\_cap, [1163](#)
- snd\_base
  - L4::Cap\_base, [894](#)
- Space\_attrb
  - L4Re::Dma\_space, [1306](#)
- ss
  - l4\_exc\_regs\_t, [1245](#)
- stack
  - L4Re::Video::View, [1502](#)
- stack.h
  - l4util\_stack\_get\_sp, [2325](#)
- start
  - L4::Kip::Mem\_desc, [1087](#)
  - L4virtio::Svr::Request\_processor, [1670](#), [1671](#)
- starts\_with
  - cxx::String, [854](#)
- State
  - L4vcpu::State, [1578](#)
- state
  - L4vcpu::Vcpu, [1591](#)
- stats\_time
  - L4::Thread, [1179](#)
- status
  - L4virtio::Svr::Dev\_config, [1636](#)
  - l4virtio\_config\_hdr\_t, [1712](#)
- Str\_cp\_in
  - L4::lpc::Str\_cp\_in, [1019](#)
- str\_cp\_in
  - L4::lpc, [659](#)
- String
  - cxx::String, [851](#)
- sub\_type
  - L4::Kip::Mem\_desc, [1087](#)
- supports
  - L4::Meta, [1110](#)
- switch\_log
  - L4::Debugger, [905](#)
- switch\_to
  - L4::Thread, [1180](#)
- symlink
  - L4Re::Vfs::Directory, [1451](#)
- sys/assert.h
  - l4\_assert, [2253](#)
- system\_shutdown
  - L4::Platform\_control, [1121](#)
- system\_suspend
  - L4::Platform\_control, [1121](#)
- tag
  - L4::lpc::Istream, [984](#)
  - L4::lpc::Ostream, [1013](#)
  - L4::lpc::Varg, [1023](#)
- take
  - L4Re::Util::Counting\_cap\_alloc, [1391](#)
  - L4Re::Util::Dataspace\_svr, [1400](#)
- Task, [519](#)
  - l4\_task\_add\_ku\_mem, [521](#)
  - l4\_task\_cap\_equal, [521](#)
  - l4\_task\_cap\_has\_child, [521](#)
  - l4\_task\_cap\_valid, [522](#)
  - l4\_task\_delete\_obj, [522](#)
  - l4\_task\_map, [523](#)
  - l4\_task\_release\_cap, [524](#)
  - l4\_task\_unmap, [524](#)
  - l4\_task\_unmap\_batch, [525](#)
  - l4\_unmap\_flags\_t, [520](#)
- task
  - L4Re::Env, [1320](#)
  - L4vcpu::Vcpu, [1591](#)
- test
  - L4::Poll\_timeout\_kipclock, [1124](#)
- Thread, [527](#)
  - l4\_thread\_arm\_set\_tpidruro, [531](#)
  - l4\_thread\_control\_flags, [529](#)
  - l4\_thread\_control\_mr\_indices, [529](#)
  - l4\_thread\_ex\_regs, [532](#)
  - l4\_thread\_ex\_regs\_flags, [531](#)
  - l4\_thread\_ex\_regs\_ret, [533](#)
  - l4\_thread\_ex\_regs\_ret\_u, [534](#)
  - l4\_thread\_ex\_regs\_u, [535](#)

- l4\_thread\_modify\_sender\_add, [536](#)
- l4\_thread\_modify\_sender\_commit, [536](#)
- l4\_thread\_modify\_sender\_start, [537](#)
- l4\_thread\_register\_del\_irq, [537](#)
- l4\_thread\_stats\_time, [538](#)
- l4\_thread\_switch, [538](#)
- l4\_thread\_vcpu\_control, [538](#)
- l4\_thread\_vcpu\_control\_ext, [539](#)
- l4\_thread\_vcpu\_control\_ext\_u, [540](#)
- l4\_thread\_vcpu\_control\_u, [541](#)
- l4\_thread\_vcpu\_resume\_commit, [542](#)
- l4\_thread\_vcpu\_resume\_start, [543](#)
- l4\_thread\_yield, [543](#)
- Thread control, [545](#)
  - l4\_thread\_control\_alien, [546](#)
  - l4\_thread\_control\_bind, [546](#)
  - l4\_thread\_control\_commit, [547](#)
  - l4\_thread\_control\_exc\_handler, [547](#)
  - l4\_thread\_control\_pager, [548](#)
  - l4\_thread\_control\_start, [548](#)
  - l4\_thread\_control\_ux\_host\_syscall, [548](#)
- Thread Control Registers (TCRs), [544](#)
- throw\_ipc\_exception
  - L4, [649](#), [650](#)
- timed\_out
  - L4::Poll\_timeout\_kipclock, [1125](#)
- timeout
  - L4::lpc\_svr::Timeout, [1055](#)
- timeout\_expired
  - L4::lpc\_svr::Timeout\_queue, [1059](#)
- Timeouts, [550](#)
  - L4\_IPC\_TIMEOUT\_0, [551](#)
  - l4\_ipc\_timeout, [552](#)
  - l4\_rcv\_timeout, [553](#)
  - l4\_snd\_timeout, [553](#)
  - l4\_timeout, [554](#)
  - l4\_timeout\_abs, [554](#)
  - l4\_timeout\_abs\_validity, [552](#)
  - l4\_timeout\_get, [555](#)
  - l4\_timeout\_is\_absolute, [556](#)
  - l4\_timeout\_rel, [556](#)
  - l4\_timeout\_rel\_get, [557](#)
  - l4\_timeout\_s, [552](#)
  - l4\_timeout\_t, [552](#)
  - l4\_utcb\_mr64\_idx, [557](#)
- Timestamp Counter, [559](#)
  - l4\_busy\_wait\_ns, [560](#)
  - l4\_busy\_wait\_us, [561](#)
  - l4\_calibrate\_tsc, [561](#)
  - l4\_get\_hz, [562](#)
  - l4\_ns\_to\_tsc, [562](#)
  - l4\_rdpmc, [563](#)
  - l4\_rdpmc\_32, [563](#)
  - l4\_rdtsc, [564](#)
  - l4\_rdtsc\_32, [564](#)
  - l4\_tsc\_init, [565](#)
  - l4\_tsc\_to\_ns, [566](#)
  - l4\_tsc\_to\_s\_and\_ns, [566](#)
  - l4\_tsc\_to\_us, [567](#)
- to\_irq
  - L4vbus::Gpio\_pin, [1554](#)
- total\_objects
  - cxx::Base\_slab, [717](#)
  - cxx::Base\_slab\_static, [722](#)
- total\_size
  - L4virtio::Virtqueue, [1692](#)
- trigger
  - L4::Triggerable, [1190](#)
- type
  - L4::lpc::Varg, [1024](#)
  - L4::Kip::Mem\_desc, [1088](#)
  - l4\_vhw\_entry, [1279](#)
  - L4Re::Vfs::Be\_file\_system, [1446](#)
  - L4Re::Vfs::File\_system, [1456](#)
- unbind
  - L4::lcu, [938](#)
  - L4::lommu, [949](#)
- Unique\_cap
  - L4Re, [672](#)
  - L4Re::Util, [687](#)
- unique\_cap
  - L4Re, [672](#)
  - L4Re::Util, [688](#)
- Unique\_del\_cap
  - L4Re, [673](#)
  - L4Re::Util, [688](#)
- unique\_del\_cap
  - L4Re, [673](#)
  - L4Re::Util, [689](#)
- unlink
  - L4Re::Namespace, [1352](#)
  - L4Re::Vfs::Directory, [1451](#)
- unlock\_all\_locks
  - L4Re::Vfs::Be\_file, [1443](#)
  - L4Re::Vfs::Generic\_file, [1465](#)
- unmap
  - L4::Task, [1171](#)
  - L4Re::Dma\_space, [1307](#)
- unmap\_batch
  - L4::Task, [1172](#)
- unmask
  - L4::lirq, [1069](#)
  - L4::lirq\_eoi, [1072](#)
- unregister\_obj
  - L4::Registry\_iface, [1135](#)
  - L4Re::Util::Object\_registry, [1426](#)
- up
  - L4::Semaphore, [1148](#)
- used\_align
  - L4virtio::Virtqueue, [1693](#)
- Used\_elem
  - L4virtio::Virtqueue::Used\_elem, [1707](#)
- used\_size
  - L4virtio::Virtqueue, [1693](#)
- utcb\_area
  - L4Re::Env, [1320](#)

- Utility Functions, [568](#)
  - [l4\\_sleep](#), [569](#)
  - [l4\\_touch\\_ro](#), [570](#)
  - [l4\\_touch\\_rw](#), [570](#)
  - [l4\\_usleep](#), [571](#)
  - [l4util\\_micros2l4to](#), [572](#)
  - [l4util\\_splitlog2\\_hdl](#), [572](#)
  - [l4util\\_splitlog2\\_size](#), [573](#)
- [ux\\_host\\_syscall](#)
  - [L4::Thread::Attr](#), [1186](#)
- [V\\_flags](#)
  - [L4Re::Video::View](#), [1499](#)
- [vCPU API](#), [623](#)
  - [L4\\_vcpu\\_state\\_flags](#), [624](#)
  - [L4\\_vcpu\\_state\\_offset](#), [624](#)
  - [L4\\_vcpu\\_sticky\\_flags](#), [625](#)
- [vCPU Support Library](#), [626](#)
  - [l4vcpu\\_irq\\_disable](#), [627](#)
  - [l4vcpu\\_irq\\_disable\\_save](#), [627](#)
  - [l4vcpu\\_irq\\_enable](#), [628](#)
  - [l4vcpu\\_irq\\_restore](#), [629](#)
  - [l4vcpu\\_is\\_irq\\_entry](#), [630](#)
  - [l4vcpu\\_is\\_page\\_fault\\_entry](#), [631](#)
  - [l4vcpu\\_print\\_state](#), [631](#)
  - [l4vcpu\\_wait\\_for\\_event](#), [632](#)
- [VM API for SVM](#), [575](#)
- [VM API for TZ](#), [576](#)
- [VM API for VMX](#), [577](#)
  - [L4\\_vm\\_vmx\\_caps\\_regs](#), [579](#)
  - [l4\\_vm\\_vmx\\_clear](#), [580](#)
  - [L4\\_vm\\_vmx\\_dfl1\\_regs](#), [579](#)
  - [l4\\_vm\\_vmx\\_field\\_len](#), [580](#)
  - [l4\\_vm\\_vmx\\_field\\_order](#), [581](#)
  - [l4\\_vm\\_vmx\\_get\\_caps](#), [582](#)
  - [l4\\_vm\\_vmx\\_get\\_caps\\_default1](#), [582](#)
  - [l4\\_vm\\_vmx\\_get\\_cr2\\_index](#), [583](#)
  - [l4\\_vm\\_vmx\\_ptr\\_load](#), [583](#)
  - [l4\\_vm\\_vmx\\_read](#), [584](#)
  - [l4\\_vm\\_vmx\\_read\\_16](#), [585](#)
  - [l4\\_vm\\_vmx\\_read\\_32](#), [586](#)
  - [l4\\_vm\\_vmx\\_read\\_64](#), [586](#)
  - [l4\\_vm\\_vmx\\_read\\_nat](#), [587](#)
  - [l4\\_vm\\_vmx\\_write](#), [587](#)
  - [l4\\_vm\\_vmx\\_write\\_16](#), [588](#)
  - [l4\\_vm\\_vmx\\_write\\_32](#), [589](#)
  - [l4\\_vm\\_vmx\\_write\\_64](#), [590](#)
  - [l4\\_vm\\_vmx\\_write\\_nat](#), [590](#)
- [Val](#)
  - [cxx::Bitfield](#), [725](#)
- [val](#)
  - [cxx::Bitfield](#), [731](#)
- [val\\_dirty](#)
  - [cxx::Bitfield](#), [731](#)
- [Val\\_unshifted](#)
  - [cxx::Bitfield](#), [725](#)
- [val\\_unshifted](#)
  - [cxx::Bitfield](#), [732](#)
- [valid](#)
  - [cxx::Bits::Base\\_avl\\_set::Node](#), [768](#)
  - [L4virtio::Svr::Virtqueue::Head\\_desc](#), [1680](#)
- [validate](#)
  - [L4::Cap\\_base](#), [895](#)
- [value](#)
  - [L4::lpc::Varg](#), [1024](#)
- [Varg\\_list\\_ref](#)
  - [L4::lpc::Varg\\_list\\_ref](#), [1027](#)
- [Vbus API](#), [593](#)
- [vbus\\_interfaces.h](#)
  - [l4vbus\\_iface\\_type\\_t](#), [2344](#)
  - [l4vbus\\_subinterface\\_supported](#), [2344](#)
- [vbus\\_types.h](#)
  - [l4vbus\\_device\\_flags\\_t](#), [2347](#)
  - [l4vbus\\_resource\\_type\\_t](#), [2347](#)
- [vcon.h](#)
  - [L4\\_vcon\\_read\\_flags](#), [2239](#)
- [vcpu\\_control](#)
  - [L4::Thread](#), [1180](#)
- [vcpu\\_control\\_ext](#)
  - [L4::Thread](#), [1181](#)
- [vcpu\\_resume\\_commit](#)
  - [L4::Thread](#), [1182](#)
- [vcpu\\_resume\\_start](#)
  - [L4::Thread](#), [1182](#)
- [version](#)
  - [l4\\_vhw\\_descriptor](#), [1276](#)
- [Video API](#), [594](#)
  - [l4re\\_video\\_goos\\_create\\_buffer](#), [596](#)
  - [l4re\\_video\\_goos\\_create\\_view](#), [597](#)
  - [l4re\\_video\\_goos\\_delete\\_buffer](#), [597](#)
  - [l4re\\_video\\_goos\\_delete\\_view](#), [599](#)
  - [l4re\\_video\\_goos\\_get\\_static\\_buffer](#), [599](#)
  - [l4re\\_video\\_goos\\_get\\_view](#), [599](#)
  - [l4re\\_video\\_goos\\_info](#), [600](#)
  - [l4re\\_video\\_goos\\_info\\_flags\\_t](#), [596](#)
  - [l4re\\_video\\_goos\\_refresh](#), [600](#)
  - [l4re\\_video\\_view\\_get\\_info](#), [601](#)
  - [l4re\\_video\\_view\\_info\\_flags\\_t](#), [596](#)
  - [l4re\\_video\\_view\\_refresh](#), [601](#)
  - [l4re\\_video\\_view\\_set\\_info](#), [601](#)
  - [l4re\\_video\\_view\\_set\\_viewport](#), [602](#)
  - [l4re\\_video\\_view\\_stack](#), [602](#)
  - [l4re\\_video\\_view\\_t](#), [595](#)
- [view](#)
  - [L4Re::Video::Goos](#), [1487](#)
- [view\\_info](#)
  - [L4Re::Util::Video::Goos\\_svr](#), [1439](#)
- [Virtual Console](#), [604](#)
  - [l4\\_vcon\\_get\\_attr](#), [607](#)
  - [l4\\_vcon\\_get\\_attr\\_u](#), [607](#)
  - [L4\\_vcon\\_i\\_flags](#), [605](#)
  - [L4\\_vcon\\_l\\_flags](#), [606](#)
  - [L4\\_vcon\\_o\\_flags](#), [606](#)
  - [l4\\_vcon\\_read](#), [608](#)
  - [l4\\_vcon\\_read\\_u](#), [609](#)
  - [l4\\_vcon\\_read\\_with\\_flags](#), [610](#)
  - [l4\\_vcon\\_send](#), [611](#)

- l4\_vcon\_send\_u, [612](#)
- l4\_vcon\_set\_attr, [613](#)
- l4\_vcon\_set\_attr\_u, [613](#)
- L4\_vcon\_size\_consts, [606](#)
- l4\_vcon\_write, [614](#)
- l4\_vcon\_write\_u, [615](#)
- Virtual Machines, [617](#)
- Virtual Registers (UTCBS), [618](#)
  - l4\_utcb\_br, [620](#)
  - l4\_utcb\_mr, [620](#)
  - l4\_utcb\_t, [619](#)
  - l4\_utcb\_tcr, [621](#)
- wait
  - L4::lpc::lstream, [985](#)
  - L4::lirq, [1070](#)
- wait\_for\_event
  - L4vcpu::Vcpu, [1592](#)
- word\_index
  - cxx::Bitmap\_base, [751](#)
- words
  - cxx::Bitmap\_base, [751](#)
- write
  - L4::Vcon, [1238](#)
  - L4virtio::Virtqueue::Desc::Flags, [1702](#)
- write\_bfm\_t
  - L4virtio::Virtqueue::Desc::Flags, [1701](#)
- writenv
  - L4Re::Vfs::Regular\_file, [1474](#)
- x86 Virtual Registers (UTCBS), [634](#)
  - L4\_utcb\_consts\_x86, [634](#)
- x86/l4/sys/cache.h, [2067](#), [2068](#)
- x86/l4/sys/consts.h, [2081](#), [2082](#)
- x86/l4/sys/l4int.h, [2174](#), [2175](#)
- x86/l4/sys/linkage.h, [1786](#), [1787](#)
- x86/l4/sys/segment.h, [1768](#), [1769](#)
  - L4\_task\_ldt\_x86\_consts, [1769](#)
- x86/l4/sys/utcb.h, [2231](#), [2233](#)
- x86/l4/util/apic.h, [1720](#), [1721](#)
- x86/l4/util/bitops\_arch.h, [1795](#), [1796](#)
- x86/l4/util/cpu.h, [1802](#), [1803](#)
- x86/l4/util/idt.h, [1728](#), [1729](#)
- x86/l4/util/irq.h, [1899](#), [1900](#)
  - l4util\_irq\_acknowledge, [1900](#)
- x86/l4/util/l4\_macros.h, [1806](#), [1807](#)
- x86/l4/util/mbi\_argv.h, [1809](#), [1810](#)
- x86/l4/util/perform.h, [1735](#), [1736](#)
- x86/l4/util/port\_io.h, [1775](#), [1777](#)
- x86/l4/util/rdtsc.h, [1746](#), [1747](#)
- x86/l4/util/spin.h, [1752](#)
- x86/l4/util/stack\_impl.h, [1813](#), [1814](#)
  - l4util\_stack\_get\_sp, [1814](#)
- x86/l4/util/util.h, [1756](#), [1758](#)
  - l4\_sleep, [1757](#)
  - l4util\_micros2l4to, [1757](#)
- x86/l4f/l4/sys/ipc.h, [2153](#), [2154](#)
- x86/l4f/l4/sys/segment.h, [1766](#), [1767](#)
- x86/l4f/l4/util/port\_io.h, [1772](#), [1775](#)
  - l4util\_ioport\_map, [1774](#)
- x86/l4f/l4/util/setjmp.h, [1782](#), [1784](#)
  - l4\_thread\_longjmp, [1783](#)
  - l4\_thread\_setjmp, [1784](#)